PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-066901
(43)Date of publication of application : 03.03.2000

(51)Int.Cl. G06F 9/45
G06F 9/318
G06F 9/34

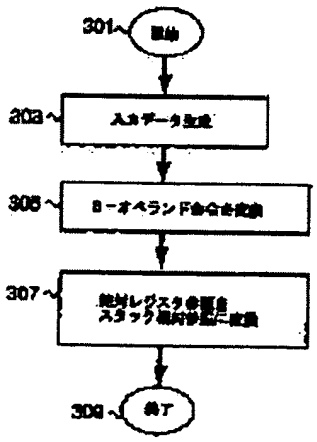
(21)Application number : 11-207155
(22)Date of filing : 22.07.1999

(71)Applicant : SUN MICROSYST INC
(72)Inventor : COX DAVID M
MOROSOV SERGUEI V
SEBERGER DAVID A
WALLACE DAVID R
WENITSKY SERGUEI L

(30)Priority
Priority number : 98 121167 Priority date : 22.07.1998 Priority country : US

(54) METHOD AND DEVICE FOR OPTIMIZING REGISTER IN STACK USING REGISTER ALLOCATING PART AND COMPUTER PROGRAM PRODUCT

(57)Abstract:
PROBLEM TO BE SOLVED: To optimize a calculation by using a register stack instead of a named register by converting a 3-operand instruction into an instruction less than a 3-operand.
SOLUTION: A stack register replacement process 300 is accessed in a compiler after a virtual register is allocated to a pseudo-register. An input data generation procedure 303 generates a control flow graph(CFG) of a related part of a compiled program, basic block representation related to the CFG and the intermediate representation(IR) of a compiled instruction. An instruction conversion procedure 305 converts an IR in a 3-operand format into an operand format less than a 3-operand. Next, a conversion procedure 307 converts absolute register reference into stack relative reference and converts pseudo-register address designation used in an operand of an IR instruction into register stack relative address designation.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－66901
（P2000－66901A）
(43)公開日　平成12年３月３日(2000.3.3)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
Ｇ０６Ｆ	9/45		Ｇ０６Ｆ	9/44	３２２Ｈ
	9/318			9/34	３４０Ａ
	9/34	３４０		9/30	３２０Ａ
				9/44	３２２Ｆ

審査請求　未請求　請求項の数44　ＯＬ　（全 27 頁）

(21)出願番号　特願平11－207155

(22)出願日　平成11年７月22日(1999.7.22)

(31)優先権主張番号　０９／１２１１６７

(32)優先日　平成10年７月22日(1998.7.22)

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595034134
サン・マイクロシステムズ・インコーポレイテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア州　94303　パロ　アルト　サン　アントニオ　ロード　901

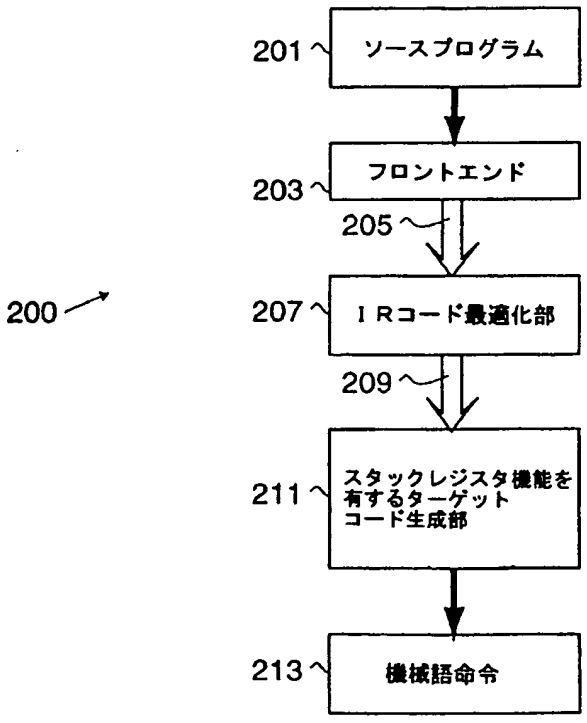
(72)発明者　ディビッド・エム・コックス
アメリカ合衆国カリフォルニア州94550・リバーモア・レグラスロード　573

(74)代理人　100089266
弁理士　大島　陽一

最終頁に続く

(54)【発明の名称】　レジスタ割当て部を用いてスタック内のレジスタを最適化するための方法、装置及びコンピュータプログラムプロダクト

(57)【要約】
【課題】　従来のレジスタ割当て技術及び最適化技術をレジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現する。
【解決手段】　コンパイラがレジスタスタックのスタックレジスタにアクセスするための効率的なコードを生成することができる装置、方法及びコンピュータプログラムプロダクトが開示される。本発明はコンパイラの中間表現内の３－オペランド命令を、１つ或いはそれ以上の３－オペランド未満の命令に変換することにより動作する。また本発明は命令のオペランドアドレス指定を、疑似名前付きレジスタへのアクセスから、レジスタスタックへのスタックオフセットを用いるスタックレジスタへのアクセスに変換する。また本発明はレジスタスタック入替えに応じて各命令においてレジスタスタック状態を判定し、それに応じてスタックレジスタへの各後続アクセスに対してスタックオフセットをマップする。



1

【特許請求の範囲】

【請求項１】　ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法であって、前記方法が、

（ａ）３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換する過程と、

（ｂ）前記１つ或いはそれ以上の３－オペランド未満の命令を解析し、１つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する過程と、

（ｃ）前記存続期間情報に応じて、ポップ動作を、前記１つ或いはそれ以上の３－オペランド未満の命令に挿入する過程と、

（ｄ）オペランド参照を、前記疑似レジスタから前記レジスタスタックへのオフセットに変換し、前記複数のスタックレジスタを参照する過程とを有することを特徴とするコンピュータ制御による方法。

【請求項２】　前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項１に記載のコンピュータ制御による方法。

【請求項３】　前記（ｄ）過程がさらに、

（ｄ１）前記疑似レジスタを前記レジスタスタックにマップする過程をさらに有することを特徴とする請求項１に記載のコンピュータ制御による方法。

【請求項４】　前記（ｄ１）過程がさらに、

（ｄ１ａ）第１の基本ブロックの開始時点でレジスタスタック状態を初期化する過程と、

（ｄ１ｂ）前記第１の基本ブロックの１つ或いはそれ以上の命令を処理し、前記疑似レジスタに対する参照を、前記複数のスタックレジスタにアクセスするための前記レジスタスタックのオフセットに置き換える過程と、

（ｄ１ｃ）必要に応じて、前記第１の基本ブロック内に１つ或いはそれ以上のレジスタスタック入替え命令を挿入する過程と、

（ｄ１ｄ）（ｄ１ｂ）過程及び（ｄ１ｃ）過程に応じて、前記レジスタスタック状態を更新する過程とを有することを特徴とする請求項３に記載のコンピュータ制御による方法。

【請求項５】　（ｄ２）前記第１の基本ブロックからの出口には存在せず、第２の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定する過程と、

（ｄ３）前記第２の基本ブロックへの入口では存在せず、前記第１の基本ブロックからの出口において現存するレジスタである一組の新規の無効レジスタを判定する過程と、

（ｄ４）前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項４に記載のコンピュータ制御による方法。

2

【請求項６】　（ｄ５）前記第１の基本ブロックと前記第２の基本ブロックとの間に正規化基本ブロックを挿入する過程と、

（ｄ６）前記正規化基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項５に記載のコンピュータ制御による方法。

【請求項７】　（ｄ５）前記第１の基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化する過程をさらに有することを特徴とする請求項５に記載のコンピュータ制御による方法。

【請求項８】　（ｄ５）前記第２の基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化する過程をさらに有することを特徴とする請求項５に記載のコンピュータ制御による方法。

【請求項９】　複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するためのコンピュータ制御による方法であって、

（ａ）前記複数のスタックレジスタに１つ或いはそれ以上の疑似レジスタを割り当てる過程と、

（ｂ）前記割り当てられた疑似レジスタを、複数のスタックオフセットを用いて前記複数のスタックレジスタにマップするレジスタスタック状態を保守する過程であって、前記レジスタスタック状態が前記スタックレジスタの変更に応答する、該過程と、

（ｃ）前記割り当てられた疑似レジスタの１つを参照する命令を変換し、前記複数のスタックオフセットの１つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの１つを識別する過程とを有することを特徴とするコンピュータ制御による方法。

【請求項１０】　前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項９に記載の方法。

【請求項１１】　前記（ｃ）過程がさらに、

３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換する過程を有することを特徴とする請求項９に記載のコンピュータ制御による方法。

【請求項１２】　（ｄ）第１の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第２の基本ブロックへの入口において存在する前記レジスタスタックに一致させる過程をさらに有することを特徴とする請求項９に記載のコンピュータ制御による方法。

3

【請求項１３】　（ｄ１）前記第１の基本ブロックに１つ或いはそれ以上の正規化命令を加える過程をさらに有することを特徴とする請求項１２に記載のコンピュータ制御による方法。

【請求項１４】　（ｄ１）前記第２の基本ブロックに１つ或いはそれ以上の正規化命令を加える過程をさらに有することを特徴とする請求項１２に記載のコンピュータ制御による方法。

【請求項１５】　（ｄ１）前記第１の基本ブロックと前記第２の基本ブロックとの間に正規化基本ブロックを加える過程と、

（ｄ２）前記正規化基本ブロックに１つ或いはそれ以上の正規化命令を加える過程とをさらに有することを特徴とする請求項１２に記載のコンピュータ制御による方法。

【請求項１６】　複数のスタックレジスタを有するレジスタスタックを用いるターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルするために、中央処理装置（ＣＰＵ）及び前記ＣＰＵに接続されるメモリを備える装置であって、

３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換するために構成される命令変換機構と、

前記１つ或いはそれ以上の３－オペランド未満の命令に含まれるオペランド参照を、１つ或いはそれ以上の疑似レジスタから前記レジスタスタックのオフセットに変換し、前記複数のスタックレジスタを参照するために構成される参照変換機構とを備えることを特徴とする装置。

【請求項１７】　前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項１６に記載の装置。

【請求項１８】　前記参照変換機構がさらに、

第１の基本ブロック内に現存する前記疑似レジスタについての情報を収集するために構成される情報収集機構と、

前記１つ或いはそれ以上の３－オペランド未満の命令を調整し、前記レジスタスタック内の現存値を保守するために構成される調整機構と、

前記１つ或いはそれ以上の３－オペランド未満の命令により参照される前記疑似レジスタを前記レジスタスタックにマップするために構成されるレジスタマップ機構とを備えることを特徴とする請求項１６に記載の装置。

【請求項１９】　前記レジスタマップ機構が、

前記第１の基本ブロックの開始時点でレジスタスタック状態を初期化するために構成されるスタック状態初期化機構と、

前記第１の基本ブロックの前記１つ或いはそれ以上の３－オペランド未満の命令を処理し、前記疑似レジスタへの参照を、前記レジスタスタック状態を用いて前記複数のスタックレジスタにアクセスするための前記レジスタ

4

スタックのオフセットに置き換えるために構成される命令処理機構と、

必要に応じて前記第１の基本ブロック内に１つ或いはそれ以上のレジスタスタック入替え命令を挿入するために構成されるレジスタスタック入替え機構と、

前記命令処理機構及び前記レジスタスタック入替え機構に対応して、前記レジスタスタック状態を更新するために構成されるレジスタスタック状態メンテナンス機構とを備えることを特徴とする請求項１８に記載の装置。

【請求項２０】　前記第１の基本ブロックからの出口において存在せず、第２の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定するために構成される現存レジスタセット判定機構と、

前記第２の基本ブロックへの入口には存在せず、前記第１の基本ブロックからの出口において存在するレジスタである一組の新規の無効レジスタを判定するために構成される無効レジスタセット判定機構と、

前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化機構とを備えることを特徴とする請求項１９に記載の装置。

【請求項２１】　前記第１の基本ブロックと前記第２の基本ブロックとの間に正規化基本ブロックを挿入するために構成される基本ブロック挿入機構と、

前記正規化基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構とをさらに備えることを特徴とする請求項２０に記載の装置。

【請求項２２】　前記第１の基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項２０に記載の装置。

【請求項２３】　前記第２の基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ利用を正規化するように構成される正規化命令挿入機構をさらに備えることを特徴とする請求項２０に記載の装置。

【請求項２４】　複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するために、中央処理装置（ＣＰＵ）及び前記ＣＰＵに接続されるメモリを備える装置であって、前記装置が、

前記複数のスタックレジスタに１つ或いはそれ以上の疑似レジスタを割り当てるために構成されるレジスタ割当て機構と、

複数のスタックオフセットを用いて前記複数のスタックレジスタに前記割り当てられた疑似レジスタをマップするレジスタスタック状態を保守するために構成されるメンテナンス機構であって、前記レジスタスタック状態が前記レジスタスタックの変更に応答する、該メンテナンス機構と、

前記割り当てられた疑似レジスタの１つを参照する命令を変換し、前記複数のスタックオフセットの１つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの１つを識別するために構成される命令参照変換機構とを備えることを特徴とする装置。

【請求項２５】　前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項２４に記載の装置。

【請求項２６】　前記命令参照変換機構がさらに、３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換するために構成される命令変換機構を備えることを特徴とする請求項２４に記載の装置。

【請求項２７】　第１の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第２の基本ブロックへの入口において存在する前記レジスタスタックに一致させるように構成される正規化機構をさらに備えることを特徴とする請求項２４に記載の装置。

【請求項２８】　前記第１の基本ブロックの１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項２７に記載の装置。

【請求項２９】　前記第２の基本ブロックに１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項２７に記載の装置。

【請求項３０】　前記第１の命令ブロックと前記第２の命令ブロックとの間に正規化基本ブロックを加えるために構成される基本ブロック挿入機構と、

前記正規化基本ブロックに１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構とをさらに備えることを特徴とする請求項２７に記載の装置。

【請求項３１】　コンピュータプログラムプロダクトであって、

複数のスタックレジスタを有するレジスタスタックを用いるターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンピュータがコンパイルできるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体を備え、前記コンピュータ読取り可能コードが、

３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記１つ或いはそれ以上の３－オペランド未満の命令のオペランド参照を、１つ或いはそれ以上の疑似レジスタから前記レジスタスタックのオフセットに変換し、前記複数のスタックレジスタを参照するために構成される参照変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とするコンピュータプログラムプロダクト。

【請求項３２】　前記参照変換機構がさらに、

第１の基本ブロック内に存在する前記疑似レジスタについての情報を収集するために構成される情報収集機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記１つ或いはそれ以上の３－オペランド未満の命令を調整し、前記レジスタスタック内の現存値を保守するために構成される調整機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記１つ或いはそれ以上の３－オペランド未満の命令により参照される前記疑似レジスタを前記レジスタスタックにマップするために構成されるレジスタマップ機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備えることを特徴とする請求項３１に記載のコンピュータプログラムプロダクト。

【請求項３３】　前記レジスタマップ機構が、

前記第１の基本ブロックの開始時点でレジスタスタック状態を初期化するために構成されるスタック状態初期化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記第１の基本ブロック内の前記１つ或いはそれ以上の３－オペランド未満の命令を処理し、前記疑似レジスタへの参照を、前記レジスタスタック状態を用いて前記複数のスタックレジスタにアクセスするための前記レジスタスタックのオフセットに置き換えるために構成される命令処理機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

必要に応じて、前記第１の基本ブロック内に１つ或いはそれ以上のレジスタスタック入替え命令を挿入するために構成されるレジスタスタック入替え機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記命令処理機構及び前記レジスタスタック入替え機構に対応して、前記レジスタスタック状態を更新するために構成されるレジスタスタック状態メンテナンス機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とする請求項３２に記載のコンピュータプ

ログラムプロダクト。

【請求項３４】　前記第１の基本ブロックからの出口において存在せず、第２の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定するために構成される現存レジスタセット判定機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記第２の基本ブロックへの入口には存在せず、前記第１の基本ブロックからの出口において現存するレジスタである一組の新規の無効レジスタを判定するために構成される無効レジスタセット判定機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとをさらに備えることを特徴とする請求項３３に記載のコンピュータプログラムプロダクト。

【請求項３５】　前記第１の基本ブロックと前記第２の基本ブロックとの間に正規化基本ブロックを挿入するために構成される基本ブロック挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記正規化基本ブロック内に１つ或いはそれ以上の正規化命令を挿入し、前記第１の基本ブロックと前記第２の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備えることを特徴とする請求項３４に記載のコンピュータプログラムプロダクト。

【請求項３６】　コンピュータプログラムプロダクトであって、

複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムをコンピュータが最適化できるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体を備え、前記コンピュータ読取り可能コードが、

前記複数のスタックレジスタに１つ或いはそれ以上の疑似レジスタを割り当てるために構成されるレジスタ割当て機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

複数のスタックオフセットを用いて前記複数のスタックレジスタに前記割り当てられた疑似レジスタをマップするレジスタスタック状態を保守するために構成されるメンテナンス機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、前記レジスタスタック状態が前記スタックレジスタの変更に応答する、該コンピュータ読取り可能プログラムコードと、

前記割り当てられた疑似レジスタの１つを参照する命令を変換し、前記複数のスタックオフセットの１つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの１つを識別するために構成される命令参照変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とするコンピュータプログラムプロダクト。

【請求項３７】　前記命令参照変換機構がさらに、３－オペランド命令を１つ或いはそれ以上の３－オペランド未満の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードを備えることを特徴とする請求項３６に記載のコンピュータプログラムプロダクト。

【請求項３８】　第１の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第２の基本ブロックへの入口において存在する前記レジスタスタックに一致させるために構成される正規化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項３６に記載のコンピュータプログラムプロダクト。

【請求項３９】　前記第１の基本ブロックに１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項３８に記載のコンピュータプログラムプロダクト。

【請求項４０】　前記第２の基本ブロックに１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項３８に記載のコンピュータプログラムプロダクト。

【請求項４１】　前記第１の基本ブロックと前記第２の基本ブロックとの間に正規化基本ブロックを加えるために構成される基本ブロック挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記正規化基本ブロックに１つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとをさらに備えることを特徴とする請求項３８に記載のコンピュー

9

タプログラムプロダクト。

【請求項４２】　ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用するためのコンピュータ制御による方法であって、前記方法により、固定レジスタ名を有する命令を生成するために用いることができる技術が、前記複数のスタックレジスタを用いる命令を生成するために用いることができ、また前記方法が、

（ａ）プッシュ或いはポップ動作を用いずに複数の命令を生成する過程であって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャにおいて実際に実装される前記複数のスタックレジスタではなく、固定名を有する１つ或いはそれ以上の疑似レジスタである、該生成過程と、

（ｂ）前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換する過程であって、それにより例えば、２つのソースオペランド及び１つのターゲットオペランドを有する３－オペランド命令が共有ソース及びターゲットを有する２－オペランド命令に変換される、該変換過程と、

（ｃ）前記複数の新規の命令を解析し、前記１つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する過程と、

（ｄ）前記存続期間情報を用いて、１つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除する過程と、

（ｅ）前記１つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化する過程と、

（ｆ）前記複数の新規の各命令に対してコンピュータ制御によるステップを実行する過程であって、前記ステップが、

（１）前記新規の命令内で、前記１つ或いはそれ以上の疑似レジスタを、前記マッピング手段を用いることにより１つ或いはそれ以上の前記複数のスタックレジスタに置き換える過程と、

（２）前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させる過程とを有する、該過程とを有し、

それにより固定レジスタ名を有する命令を生成するために用いることができる前記技術を用いて、前記複数のスタックレジスタを用いる命令を生成できることを特徴とするコンピュータ制御による方法。

【請求項４３】　ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用するために、中央処理装置（ＣＰＵ）及び前記ＣＰＵに接続されるメモリを備える装置であって、前記装置により、固定レジスタ名を有する命令を生成するために用いることができる技術が、前記複数

10

のスタックレジスタを用いる命令を生成するために用いることができ、また前記装置が、

プッシュ或いはポップ動作を用いずに複数の命令を生成するために構成される命令生成機構であって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャ内に実際に実装される前記複数のスタックレジスタではなく、固定名を有する１つ或いはそれ以上の疑似レジスタである、該命令生成機構と、

前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換するために構成される命令変換機構であって、それにより例えば２つのソースオペランドと１つターゲットオペランドを有する３－オペランド命令が共有ソース及びターゲットを有する２－オペランド命令に変換される、該命令変換機構と、

前記複数の新規の命令を解析し、前記１つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定するために構成される命令解析機構と、

前記存続期間情報を用いて、１つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除するために構成される無効値削除機構と、

前記１つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化するために構成される初期化機構と、

前記複数の新規の各命令を変換するために構成される疑似レジスタ変換機構とを備え、前記疑似レジスタ変換機構が、

前記新規の命令内において、前記１つ或いはそれ以上の疑似レジスタを前記マッピング手段を用いることにより１つ或いはそれ以上の前記複数のスタックレジスタに置き換えるために構成されるオペランド置換え機構と、

前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させるために構成される調整機構とを備え、

それにより前記装置が、前記複数のスタックレジスタを用いる命令を生成するために、固定レジスタ名を有する命令を生成するために用いることができる前記技術を組み込むことを特徴とする装置。

【請求項４４】　固定レジスタ名を有する命令を生成するために用いることができる技術が、複数のスタックレジスタを用いる命令を生成するために用いられるようにするコンピュータプログラムプロダクトであって、

コンピュータがターゲットコンピュータアーキテクチャ内に前記複数のスタックレジスタを備えるレジスタスタックを効率的に用いることができるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体であって、前記コンピュータ読取り可能コードが、

プッシュ或いはポップ動作を用いずに複数の命令を生成するために構成される命令生成機構を前記コンピュータ

が実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャ内に実際に実装される前記スタックレジスタではなく、固定名を有する１つ或いはそれ以上の疑似レジスタである、該コンピュータ読取り可能プログラムコードと、

前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、それにより例えば２つのソースオペランドと１つのターゲットオペランドとを有する３－オペランド命令が、共有ソース及びターゲットを有する２－オペランド命令に変換される、該コンピュータ読取り可能プログラムコードと、

前記複数の新規の命令を解析し、前記１つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定するために構成される命令解析機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記存続期間情報を用いて、１つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除するために構成される無効値削除機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記１つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化するために構成される初期化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記複数の新規の各命令を変換するために構成される疑似レジスタ変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有し、前記疑似レジスタ変換機構が、

前記新規の命令内において、前記１つ或いはそれ以上の疑似レジスタを前記マッピング手段を用いて１つ或いはそれ以上の前記複数のスタックレジスタに置き換えるために構成されるオペランド置換え機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させるために構成される調整機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備え、

それにより前記プロダクトが、前記複数のスタックレジスタを用いる命令を生成するために、固定レジスタ名を有する命令を生成するために用いることができる前記技術を組み込むことを特徴とするコンピュータプログラムプロダクト。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はコンピュータシステム用の最適化コンパイラの分野に関連する。詳細には、本発明は名前付レジスタの代わりに、レジスタスタックを用いて計算を最適化するための新規で、有用な最適化方法、装置及びコンピュータプログラムプロダクトに関連する。

【０００２】

【従来の技術】最近の多くのコンピュータアーキテクチャは、特定レジスタを識別するために名前を用いる。最適化コンパイラは、そのような名前付レジスタの使用を最適化するための多くの技術を発展させた。いくつかのコンピュータアーキテクチャでもレジスタに対してレジスタスタックを用いる。インテル製プロセッサは、浮動小数点演算に対してレジスタスタックを用いる。このレジスタスタック構成は、Intel Architecture Software Developer's Manual: Basic Architecture (order number 243190, ｸ1996, 1997) の第７章に記載されており、その部分を参照して本明細書の一部としている。

【０００３】インテル製ＦＰＵに対する浮動小数点最適化技術は、Inter Architecture Optimization Manual (order number 242816-003, ｸ1996, 1997) の第５章に記載されており、その部分を参照して本明細書の一部としている。

【０００４】コンパイラレジスタ割当て技術は、固定名を有し、スタックレジスタを用いないレジスタを取り扱うことにより主に発展した。演算がレジスタスタック上で実行される際に、スタックレジスタ内に保持される値がレジスタスタック内を移動するため、この現存の技術はレジスタスタックを取り扱わない。

【０００５】

【発明が解決しようとする課題】固定名レジスタを取り扱う従来のレジスタ割当て技術及び最適化技術を、レジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現する。

【０００６】

【課題を解決するための手段】本発明は、レジスタスタックを含むターゲットコンピュータに配向されるコンピュータ命令を生成する最適化コンパイラを改善する。本発明は、コンパイラの最適化段階の多くの場合に、固定名を有する疑似レジスタを用いる方法を開示する。その後段階では、疑似レジスタを用いる命令は変更され、代わりにスタックレジスタが用いられる。スタック動作のプッシュ及びポップを実行する命令の動作により生じるレジスタスタックの入替えは、疑似レジスタをスタックレジスタにマップするレジスタスタック状態により追跡される。スタックレジスタは、レジスタスタックへのオ

フセットにより参照される。

【０００７】本発明の一態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法である。その方法はプッシュ或いはポップ動作を用いない複数の命令を生成し、そのオペランドは、ターゲットコンピュータアーキテクチャ内に実際に実装される複数のスタックレジスタではなく、固定名を有する１つ或いはそれ以上の疑似レジスタである。またその方法は複数の命令を複数の新しい命令に変換する。複数の新しい命令は、ターゲットコンピュータアーキテクチャの命令のフォーマットに対応するオペランドフォーマットを有する。例えば、２つのソースオペランド及び１つのターゲットオペランドを有する３－オペランド命令は、共有のソース及びターゲットを有する２－オペランド命令に変換される。またその方法は複数の新しい命令を解析し、１つ或いはそれ以上の疑似レジスタに対する存続期間情報を確定する過程を含む。その方法は存続期間情報を用いて、１つ或いはそれ以上のポップ動作を複数の新しい命令に導入することにより無効な値を削除する。さらにその方法は１つ或いはそれ以上の疑似レジスタから複数のスタックレジスタへのマッピング手段を初期化する。またその方法は、複数の新しい命令のそれぞれに対して、以下のコンピュータ制御による過程を実行する。その過程は、新しい命令内でマッピング手段を用いることにより、１つ或いはそれ以上の疑似レジスタを１つ或いはそれ以上の複数のスタックレジスタに置き換える過程と、マッピング手段を更新してスタック効果を新しい命令のマッピングに反映させる過程とを含む。このようにその方法は、固定レジスタ名を有する命令を生成するための既存技術を用いて、複数のスタックレジスタを用いる命令を生成する。

【０００８】本発明の別の態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるために、中央処理装置（ＣＰＵ）及びそのＣＰＵに接続されるメモリを備える装置である。その装置は、プッシュ或いはポップ動作を用いない複数の命令を生成するために構成された命令生成機構を備え、そのオペランドは、ターゲットコンピュータアーキテクチャ内に実際に実装される複数のスタックレジスタの代わりに、固定名を有する１つ或いはそれ以上の疑似レジスタである。またその装置は、複数の命令を複数の新しい命令に変換するために構成された命令変換機構を備える。複数の新しい命令は、ターゲットコンピュータアーキテクチャの命令フォーマットに対応するオペランドフォーマットを有する。例えば２つのソースオペランド及び１つのターゲットオペランドを有する３－オペランド命令は、共有ソース及びターゲットを有する２－オペランド命令に変換される。さらにその装置は、複数の新しい命令を解析し、１つ或いはそれ以上の疑似レジスタに対する存続期間情報を確定するために構成された命令解析機構を備える。存続期間情報を用いて１つ或いはそれ以上のポップ動作を複数の新しい命令に導入するために構成された無効値削除機構により、無効な値は削除される。さらにその装置は、１つ或いはそれ以上の疑似レジスタから複数のスタックレジスタへのマッピング手段を初期化するために構成された初期化機構を備える。またその装置は、複数の新しい命令をそれぞれ変換するために構成された疑似レジスタ変換機構を備える。疑似レジスタ変換機構はさらに、新しい命令の内でマッピング手段を用いて、１つ或いはそれ以上の疑似レジスタを１つ或いはそれ以上の複数のスタックレジスタに置き換えるために構成されたオペランド置換え機構と、マッピング手段を更新してスタック効果を新しい命令のマッピングに反映させるために構成された調整機構とを備える。このようにしてその装置は、固定レジスタ名を有する命令を生成するために用いられる既存技術を組み込み、複数のスタックレジスタを用いる命令を生成する。

【０００９】本発明の別の態様はコンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体はその中に組み込まれるコンピュータ読取り可能コードを備える。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、そのコンピュータはターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用できるようになる。コンピュータ上で実行される際に、そのコンピュータ読取り可能コードにより、コンピュータは命令変換機構、命令解析機構、無効値削除機構、初期化機構、疑似レジスタ変換機構、オペランド置換え機構及び調整機構を実現できるようになる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【００１０】本発明の別の態様は、ターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するためのコンピュータ制御による方法である。ターゲットコンピュータアーキテクチャは複数のスタックレジスタを備えるレジスタスタックを含む。その方法は、１つ或いはそれ以上の疑似レジスタを複数のスタックレジスタに割り当てる過程を有する。別の過程は、複数のスタックオフセットを用いて、その割り当てられた疑似レジスタを複数のスタックレジスタにマップするレジスタスタック状態を保守する過程である。レジスタスタック状態はレジスタスタックの変化に応答する。またその方法は、１つの割り当てられた疑似レジスタを参照する命令を、複数のスタックオフセットの１つを用いる命令に変換する。複数のスタックオフセットの１つを用いるために命令を変換することにより、その命令は、レジスタスタック状態により複数のスタックレジスタの１つを識別できるようになる。

15

【００１１】本発明のさらに別の態様は、ターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するために、中央処理装置（ＣＰＵ）及びＣＰＵに接続されるメモリを備える装置である。ターゲットコンピュータアーキテクチャは、複数のスタックレジスタを備えるレジスタスタックを含む。その装置は、１つ或いはそれ以上の擬似レジスタを複数のスタックレジスタに割り当てるために構成されるレジスタ割当て機構を備える。さらにその装置は、レジスタスタック状態を保守するために構成されるメンテナンス機構を備える。レジスタスタック状態は複数のスタックオフセットを用いて、その割り当てられた疑似レジスタを、複数のスタックレジスタにマップする。レジスタスタック状態は、レジスタスタックの変化に応答する。またその装置は、割り当てられた疑似レジスタの１つを参照する命令を、複数のスタックオフセットの１つを用いる命令に変換するために構成される命令参照変換機構を備える。変換された命令オペランドは、レジスタスタック状態により複数のスタックレジスタの１つを識別する。

【００１２】本発明の別の態様は、コンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体はその中に組み込まれるコンピュータ読取り可能コードを有する。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータは、複数のスタックレジスタを備えるレジスタスタックを含むターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化できるようになる。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータはレジスタ割当て機構、メンテナンス機構及び命令参照変換機構を実現することができる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【００１３】本発明の別の態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法である。その方法は、３－オペランド命令を、１つ或いはそれ以上の３－オペランド命令未満の命令に変換する過程を有する。またその方法は１つ或いはそれ以上の３－オペランド命令未満の命令を解析し、１つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する。さらにその方法は、存続期間情報に応じて、ポップ動作を１つ或いはそれ以上の３－オペランド命令未満の命令に挿入する過程を有する。オペランドアドレスは、割り当てられた疑似レジスタから、レジスタスタックのオフセットに変換され、複数のスタックレジスタをアドレス指定する。

【００１４】本発明の別の態様は、ターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルするために、中央処理装置（ＣＰＵ）及びＣＰＵに接続されるメモリを備える装置である。ターゲットコンピュータアーキテクチャは、複数のスタックレジスタを備えるレジスタスタックを用いる。その装置は、３－オペランド命令を１つ或いはそれ以上の３－オペランド命令未満の命令に変換するために構成される命令変換機構を備える。またその装置は、１つ或いはそれ以上の疑似レジスタからの１つ或いはそれ以上の３－オペランド命令未満の命令に含まれるオペランドアドレスをレジスタスタックのオフセットに変換し、複数のスタックレジスタをアドレス指定するために構成されるアドレス変換機構を備える。

16

【００１５】本発明のさらに別の態様は、コンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体がその内部にコンピュータ読取り可能コードを有する。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータはターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルすることができる。ターゲットコンピュータアーキテクチャは複数のスタックレジスタを備えるレジスタスタックを用いる。コンピュータ上で実行される際に、コンピュータ読取り可能コードによりコンピュータは、命令変換機構及びアドレス変換機構を実現することができる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【００１６】生変数解析（live-variable analysis）の議論は、「Compilers: Principles, Techniques and Tools by Alfred V」（Aho, Ravi Sethi and Jeffrey D. Ullman, Addison-Wesley Publishing Co. 1988, ISBN 0-201-10088-6）の６０８～６３３ページに見い出すことができ、ここで参照して本明細書の一部としている。

【００１７】本発明の上記態様及び多くの他の態様は、種々の図面において例示される以下の好適の実施例の詳細な説明から当業者には明らかになるであろう。

【００１８】

【発明の実施の形態】本発明はコンピュータを用いる。概ね参照番号１００を付され、本発明を支援するように構成されるコンピュータのいくつかの構成要素が図１に示されており、プロセッサ１０１は中央処理装置（ＣＰＵ）１０３、メモリセクション１０５及び入力／出力（Ｉ／Ｏ）セクション１０７を備えている。Ｉ／Ｏセクション１０７はキーボード１０９、ディスプレイユニット１１１、ディスク記憶ユニット１１３及びＣＤ－ＲＯＭドライブユニット１１５に接続される。ＣＤ－ＲＯＭドライブユニット１１５は、ＣＤ－ＲＯＭ媒体１１７を読み出すことができ、ＣＤ－ＲＯＭ媒体１１７は典型的にはプログラム及びデータ１１９を含む。メモリセクション１０５は、コンピュータ１００が本発明の過程を実行できるようにする実行プログラム１２１を含む。またそのコンピュータは、スタックオフセットを介してアド

レス指定されるいくつかのスタックレジスタを含むレジスタスタック１２３を備える場合もある。好適な実施例では、レジスタスタック１２３はインテル製浮動小数点ユニットアーキテクチャにより動作する。ディスク記憶ユニット１１３及びＣＤ－ＲＯＭ媒体１１７を備えるＣＤ－ＲＯＭドライブユニット１１５はファイル記憶機構を備える。そのようなコンピュータシステムは、本発明を実現する応用例を実行することができる。図１に示されるコンピュータシステムが、本発明の各実施例に対して必要とされないこともある装置を含むことは当業者には理解されよう。詳細にはレジスタスタック１２３は、実行プログラム１２１を含む同じコンピュータ内に備えられる必要はない。

【００１９】図２は概ね参照番号２００により示され、本発明を用いるコンパイラアーキテクチャを示す。コンパイラアーキテクチャ２００により実装されるコンパイラは、コンパイラフロントエンドセグメント２０３によりターゲットプログラムのソース情報２０１を使用する。コンパイラフロントエンドセグメント２０３は、ターゲットプログラムのソース情報２０１に適用可能なプログラミング言語の規則によりターゲットプログラムのソース情報２０１の構文及び意味を処理する。コンパイラフロントエンドセグメント２０３は、ターゲットプログラムのソース情報２０１の「中間」コード表現２０５の少なくとも１つのバージョンを生成する。中間表現（ＩＲ）は概ね、制御フローグラフ（ＣＦＧ）を表すか、或いは制御フローグラフを作成するために用いることができるデータ構造を備える。ＣＦＧを用いて、プログラムの基本ブロック間の制御の流れを表す。基本ブロックは、制御フロー構造、関数、手順或いは基本ブロック内の実行の流れを変更する他の構成体を全く含まない。基本ブロックは、１つの入口点及び１つの出口点のみを備える。

【００２０】その後、「中間」コード表現２０５は、中間表現最適化部セグメント２０７により最適化される。中間表現最適化部セグメント２０７は、ターゲットプログラムのソース情報２０１の「中間」コード表現２０５上で動作及び調整し、当業者には周知の種々の方法によりプログラムの実行を最適化する。中間表現最適化部セグメント２０７は最適化済み中間表現２０９を生成する。コード生成部セグメント２１１（本発明の態様を含む）は、その最適化済み中間表現２０９を用いて、低レベルの最適化を実行し、物理的なレジスタを割り当て、さらにその最適化済み中間表現２０９からアセンブラソースコード並びにまたオブジェクトコードモジュール２１３を生成する。オブジェクトコードは、オブジェクトモジュールにおける２値コンピュータ命令（オペコード）を備える。アセンブラソースコードは、アセンブラソース言語における一連の記号ステートメントである。アセンブラソースコード及びオブジェクトコードのいずれも、特定のコンピュータアーキテクチャ（例えばインテル製Ｐｅｎｔｉｕｍアーキテクチャ）を目的とする。

【００２１】上記のように、コード生成部セグメント２１１は低レベルの最適化を実行し、オブジェクトコード（オブジェクトモジュールの形式をなす場合が多い）或いはアセンブラソースコードのいずれか（或いは両方）を生成する。プログラムの最適化済み中間表現２０９は概ね、固定名を有する仮想レジスタを参照する。すなわち中間表現最適化部セグメント２０７はターゲットコンピュータが無限の名前付き仮想レジスタを含むことを仮定する。コード生成部セグメント２１１の動作中に、これらの仮想レジスタはターゲットコンピュータの物理的レジスタに割り当てられる。このリソース管理は、レジスタ割当て段階或いはプロセスによりコード生成部セグメント２１１において実行される。レジスタ割当てプロセスの一態様は、物理的レジスタの内容がプログラム実行中の種々の時点でメモリに「流出」されることが多いが、限られた数の物理的レジスタを用いて、その種々の時点でのプログラムにより適した値を保守できるようにするということである。そのメモリに流出した値は、プログラムが種々の実行段階まで進む際に、物理的レジスタに復元される場合が多い。レジスタ割当てプロセスは、いずれの物理的レジスタがその基本ブロックに用いられるかを判定する。

【００２２】最適化コンパイラ及び関連する技術の一般的な議論は、「Compilers: Principles, Techniques and Tools by Alfred V. Aho」（Ravi Sethi and Jeffrey D.Ullman, Addison-Wesley Publishing Co. 1988, ISBN 0-201-10088-6）、詳細には４６３－７２３ページの第８、９及び１０章に見い出すことができ、ここで参照して本明細書の一部としている（これ以降Ahoとよぶ）。レジスタ割当て及びメモリへの値の流出に関連するさらなる情報は、「Register Allocation & Spilling via Graph Coloring」（G.J. Chaitin, ｸ1982, Proceedings of the SIGPLAN '82 Symp. On Compiler Construction, June 1982, pp. 98-105）により提供され、ここで参照して本明細書の一部としている。

【００２３】ここではいくつかのマシンがスタックとして構成される物理的レジスタを備え、これらのレジスタは固定名を用いて参照されることができない。

【００２４】本発明の一態様は、固定名を有する一組の疑似レジスタが、レジスタ割当て段階まで、さらにレジスタ割当て段階を通して用いられ、これらの疑似レジスタが、実在の物理的レジスタであるかのように取り扱われるということである。このようにしてコンパイラの中間表現最適化段階は現存する機構を用いて、その疑似レジスタが後にスタックレジスタに置き換えられるかには関係なく疑似レジスタを最適化する。このようにして従来のレジスタ割当て及び流出動作（及び多くの他のコード生成及び最適化動作）が疑似レジスタに適用される。

疑似レジスタの数はスタックレジスタの数と同じである。スタックレジスタによる疑似レジスタの実際の置換えは、コンパイラがレジスタ割当てを実行した後に呼び出される命令の状況により達成される。

【００２５】好適な実施例は、インテル製ＦＰＵアーキテクチャ（或いはその相当品）を用いるターゲットコンピュータに向けられる。インテル製ＦＰＵアーキテクチャは、２－オペランドまでのオペランド及び浮動小数点レジスタスタックを用いるコンピュータ浮動小数点構成を用いる。本発明を用いて、任意のレジスタスタック内のレジスタへのアクセスを最適化することができるが、それは浮動小数点アーキテクチャに制限されないことは、添付の図面及びその説明から当業者には理解されよう。

【００２６】以下の説明は本発明の動作の概要を示す。後続する説明はその動作の細部に及ぶ。これらの説明はインテル互換性ＦＰＵアーキテクチャに向けられる。ここで開示される発明は他のレジスタスタック用アーキテクチャにも適用できることは当業者には理解されよう。

【００２７】図３は概ね参照番号３００により示され、疑似レジスタをレジスタスタック内のスタックレジスタに割り当てるためのスタックレジスタ置換えプロセスの概要を与える。スタックレジスタ置換えプロセス３００は、仮想レジスタが疑似レジスタに割り当てられた後にコンパイラにおいて呼び出される。

【００２８】スタックレジスタ置換えプロセス３００は、「開始」位置３０１で開始し、「入力データ生成」手順３０３に続く。「入力データ生成」手順３０３は、コンパイルされているプログラムの関連部分の制御フローグラフ（ＣＦＧ）、ＣＦＧに関連する基本ブロック表現及びコンパイル済み命令の中間表現を生成する。入力データのいくつか或いは全部が、コンパイラの上記部分により生成されることができる。次にＩＲは、３－オペランド形式（概ね２ソースオペランド及び１ターゲットオペランドを有する）からのＩＲを、インテル製アーキテクチャの命令フォーマットに対応する３－オペランド未満のオペランド形式に変換する「命令変換」手順３０５により変更される。「命令変換」手順３０５は後に図４に関連して記載される。次に「絶対レジスタ参照をスタック相対参照に変換」手順３０７は、ＩＲ命令のオペランドにおいて用いられる疑似レジスタアドレス指定を、レジスタスタック相対アドレス指定に変換する。「絶対レジスタ参照をスタック相対参照に変換」手順３０７は後に図６に関連して記載される。最終的にスタックレジスタ置換えプロセス３００は「終了」位置３０９を通り終了する。

【００２９】図４は概ね参照番号４００により示され、ＩＲの３－オペランド形式をより少ないオペランドを有するインテルフォーマット命令に変換するためのＩＲ命令変換プロセスを示す。プロセス４００は、図３の「命令変換」手順３０５により呼び出される。プロセス４００は「開始」位置４０１で開始し、「各基本ブロックを巡回」手順４０３に続く。「各基本ブロックを巡回」手順４０３はＣＦＧの各基本ブロックを巡回する（繰り返す）。繰返し済み基本ブロックはそれぞれ、図５に関して後に記載される「３－オペランドを変換」手順４０５により処理される。各基本ブロックを巡回後、プロセス４００は「終了」位置４０７を通り終了する。

【００３０】ＩＲ命令は概ね、それぞれ異なることができる２つのソースオペランド及び１つのターゲットオペランドを有する３－オペランドフォーマットにおいて存在する。従ってインテル互換性浮動小数点アーキテクチャの場合、図５に関連して記載されるように各基本ブロックのＩＲは変更され、任意の３－オペランド浮動小数点ＩＲ命令は、３－オペランド未満のオペランドを備えるインテル命令フォーマットを有するＩＲ命令に、或いは３－オペランド未満のオペランドを有する一連のＩＲ命令に変換する。３－オペランド命令は、行先（デスティネーション）とは無関係であることができる２つのソースを識別する命令である（例えば、ｏｐＲ１，Ｒ２→Ｒ３、ここでｏｐが減算である場合には、Ｒ２はＲ１から引かれ、その結果がＲ３に格納される）。インテルフォーマット２－オペランド命令は２つのソースを用いており、ソースレジスタの１つに結果を残し、その元の値を消失させる（例えば、ｏｐＲ１，Ｒ２、ここでｏｐが減算である場合には、Ｒ２はＲ１から引かれ、その結果がＲ１に残る）。いくつかの命令（例えばインテル浮動小数点平方根命令）は１つのソース上で動作し、その結果（例えばｏｐＲ１）を有するソースを上書きする。ＩＲ３－オペランド命令は、移動命令及び２－オペランド命令（例えばｏｐＲ１，Ｒ２→Ｒ３は、ｍｏｖｅＲ３，Ｒ１；ｏｐＲ３，Ｒ２に変換する）を用いることによりインテルフォーマットに変換されることができる。図５はこれらの変換がいかにして実現されるかを示す。

【００３１】図５は概ね参照番号４２０により示され、図４の「３－オペランドを変換」手順４０５により呼び出されるＩＲ命令変換プロセスを示す。プロセス４２０は「開始」位置４２１で開始し、「ＩＲ命令繰返し」手順４２３に続く。「ＩＲ命令繰返し」手順４２３は、基本ブロックの各命令を繰り返す。基本ブロックの全ての命令が処理された場合、プロセス４２０は「終了」位置４２５を通り終了する。

【００３２】その基本ブロック（基本ブロックは「各基本ブロック巡回」手順４０３により繰り返された）の各ＩＲ命令は、ＩＲ命令の指数部を取得する「命令指数部確定」手順４２７により検査される。これらの指数部は命令のタイプ、ＩＲ命令により用いられるオペランド数及びソース並びにデスティネーションオペランドの位置を含む。次に「関連命令」決定手順４２８により「命令指数部確定」手順４２７により確定される指数部が検査

され、命令タイプが、ＩＲ命令が疑似レジスタの１つを参照しない（すなわちＩＲ命令が関連しない）ということを示す場合には、プロセス４２０は「ＩＲ命令繰返し」手順４２３に戻り、次のＩＲ命令を処理する。しかしながらＩＲ命令が関連する（すなわち疑似レジスタの１つを参照する）場合には、プロセス４２０は「指数部選択」手順４２９に続く。

【００３３】「指数部選択」手順４２９はＩＲ命令オペコード及びそのオペランドを検査し、可能なＩＲ命令変換を確定する。プログラム順（in-order）選択手順４３１は、ＩＲ命令が命令の第１のオペランドにその結果を格納するか否かを判定する。この条件が成り立つ場合には、プロセス４２０は、命令の３－オペランドを３－オペランド未満の適当なインテルフォーマットに変換する「基本変換」手順４３２に続く。その後プロセス４２０は、「ＩＲ命令繰返し」手順４２３の繰返しを継続する。

【００３４】「実行順序変換（out-of-order）、交換可能」選択手順４３３は、ＩＲ命令が命令の第２のオペランドにその結果を格納するか否か、並びにその演算が交換可能であるか否かを判定する。そうである場合には、プロセス４２０は、命令のオペランドの順序を交換すると共に、命令の３－オペランド形式を、３－オペランド形式未満の適当なインテルフォーマットに変換する「オペランド順序交換」手順４３５に続く。その後プロセス４２０は、「ＩＲ命令繰返し」手順４２３の繰返しを継続する。

【００３５】「実行順序変換、交換不可、両方向ｏｐ」選択手順４３７は、ＩＲ命令が命令の第２のオペランドにその結果を格納するか否か、並びにその演算が交換不可であるか否かを判定する。そうである場合には、プロセス４２０は、「オペコードを逆向きオペコードに置換え」手順４３９に続き、その手順４３９が命令の３－オペランド形式を、適当な逆向き２－オペランド形式に変換する（例えば、「subtract Ｒ１,Ｒ２→Ｒ２」（Ｒ１からＲ２を引いて、その結果をＲ２に残す）は、「rsubtract Ｒ２，Ｒ１」に変換される）。その後プロセス４２０は「ＩＲ命令繰返し」手順４２３の繰返しを継続する。

【００３６】「デフォルト」手順４４１は、上記ケースが適用されない場合に用いられる。この状況では、プロセス４２０は、適当なソースを行先に複製し、その後複製値にその動作を適用する「命令拡張」手順４４３に続く。その後プロセス４２０は「ＩＲ命令繰返し」手順４２３の繰返しを継続する。

【００３７】このようにしてインテルフォーマット２－オペランド命令を用いるターゲットコンピュータアーキテクチャの場合に、プロセス４２０は、ＩＲ３－オペランド形式からの全ての関連する命令を、３－オペランド形式未満のインテルフォーマットに変換する。ＩＲ命令を、インテル互換性ＦＰＵと共に用いられるＦＳＱＲＴ命令のような１オペランド命令に変換する方法は当業者には理解されよう。

【００３８】図６は概ね参照番号５００により示され、疑似レジスタを参照するＩＲ命令オペランドを、レジスタスタックへのスタックオフセットを用いてスタックレジスタを参照するＩＲ命令オペランドに変換するための「絶対レジスタ参照をスタック相対参照に変換」プロセスを示す。プロセス５００は、図３の「絶対レジスタ参照をスタック相対参照に変換」手順３０７により呼び出される。プロセス５００は「開始」位置５０１で開始し、図７に関連して後に記載されるように、疑似レジスタの使用についての存続期間情報を収集する「疑似レジスタについての情報を収集」手順５０３に続く。存続期間情報は、レジスタのその値が無効になる時点を判定する。次にプロセス５００は関連するＩＲ命令（すなわち疑似レジスタにアクセスする命令）を検査し、実行時に、レジスタスタックから無効レジスタを除去する命令を挿入或いは変更する。「スタックから無効レジスタ除去」手順５０５は図８に関連して後に記載される。次にプロセス５００は、疑似レジスタを参照するオペランドを、レジスタスタック内のスタックレジスタを参照するオペランドに置き換える「疑似レジスタをスタックレジスタ上にマップ」手順５０７に続く。「疑似レジスタをスタックレジスタ上にマップ」手順５０７は図７に関連して後に記載される。プロセス５００は「終了」位置５０９を通り終了する。

【００３９】図７は概ね参照番号５１０により示され、図６の「疑似レジスタについての情報を収集」手順５０３により呼び出される、疑似レジスタ存続期間伝搬プロセスを示す。プロセス５１０は、ＣＦＧ及び関連する基本ブロックにより表されるプログラムの疑似レジスタ存続期間を伝搬する。またプロセス５１０は、疑似レジスタが各基本ブロックへの入口及び基本ブロックからの出口上に存在する情報を判定及び退避する。この情報を用いて、レジスタの値が存在する領域を判定する。レジスタが存在しない場合には、「無効（dead）」と呼ばれる。存続期間解析は当分野において周知であり、「Ａｈｏ」に記載される。

【００４０】「ＣＦＧの疑似レジスタ最適化」手順５１３は、いずれの疑似レジスタが各基本ブロックの入口に存在するかを判定し、ｉｎｐｕｔ（ｂｌｏｃｋ＿ｉｄ）としてこの情報を退避する。またこの手順５１３は、いずれの疑似レジスタが各基本ブロックの出口に存在するかを判定し、この情報をｏｕｔｐｕｔ（ｂｌｏｃｋ＿ｉｄ）として退避する。このプロセス５１０は「終了」位置５２１を通り終了する。

【００４１】図８は概ね参照番号５３０により示され、無効疑似レジスタに関連するスタックレジスタが、無効である場合にレジスタスタックからポップされるように

関連するＩＲ命令を変更するための「スタックから無効レジスタを除去」プロセスを示す。プロセス５３０は、図６の「スタックから無効レジスタを除去」手順５０５から呼び出され、「開始」位置５３１で開始する。プロセス５３０は、ＣＦＧの各基本ブロックを巡回する「各基本ブロック巡回」手順５３３に続く。プロセス５３０は、各基本ブロックを、疑似レジスタの１つにアクセスする基本ブロックの各命令を繰り返す「基本ブロックの各関連命令繰返し」手順５３５に渡す。一度基本ブロックの全ての命令が繰り返された場合には、プロセス５３０は「各基本ブロック巡回」手順５３３に戻り、次の基本ブロックを処理する。「基本ブロックの各関連命令繰返し」手順５３５により繰り返される命令は、ＩＲ命令を変更し、スタックから無効スタックレジスタを除去する「スタック安定化」手順５３６に渡される。これはオペレーションコードを、スタックポップを実行し、スタックから無効な値を除去するバージョンに置き換えることにより行われる。また、この時点でＩＲ命令はレジスタスタックに直接アクセスしないということは当業者には理解されよう。代わりにＩＲ命令は、レジスタスタック内のスタックレジスタにマップされることになる疑似レジスタにアクセスする。「スタック安定化」手順５３６はさらに図１０に関連して記載される。全ての基本ブロックが処理された後、プロセス５３０は「終了」位置５３７を通り終了する。命令のオペランドが疑似レジスタを表し、スタックレジスタを表さない場合でも、レジスタスタックに作用するためのＩＲ命令の変更を、この時点で行なうことができることは当業者には理解されよう。

【００４２】図９は概ね参照番号５５０により示され、関連命令のオペランドを、疑似レジスタ参照を用いることからレジスタスタックオフセット参照を用いることに変換する「疑似レジスタアドレス指定をレジスタスタックアドレス指定に変換」プロセスを示す。またプロセス５５０は、ＣＦＧのエッジにより接続される基本ブロック間のレジスタスタックを正規化する。プロセス５５０は図６の「疑似レジスタをスタックレジスタ上にマップ」手順５０７により呼び出され、「開始」位置５５１で開始する。プロセス５５０は、ＣＦＧの先頭から「スタックレジスタをマップ」手順５５５に各基本ブロックを与えるＣＦＧの底までＣＦＧを通過する「先頭から底までＣＦＧを通過」手順５５３に続く。「スタックレジスタをマップ」手順５５５は、各関連する命令のオペランドを、疑似レジスタの代わりに、レジスタスタックオフセットに変換する。「スタックレジスタをマップ」手順５５５はさらに図１１に関連して記載される。

【００４３】一度関連する命令が各基本ブロックにおいて変換されたなら、プロセス５５０は、ＣＦＧをその底からその先頭まで通過し、ＣＦＧの各エッジ及びそのエッジに関連する基本ブロックにおいて「基本ブロック正規化」手順５５９を呼び出す「底から先頭までＣＦＧを通過」手順５５７に続く。「基本ブロック正規化」手順５５９は、レジスタスタックを入れ替える命令を、適当な基本ブロック（或いは新規に作成された基本ブロック）に挿入することにより達成される。「基本ブロック正規化」手順５５９は後に図１２及び図１３に関連して記載される。「底から先頭までＣＦＧを通過」手順５５７が終了した後、プロセス５５０は「終了」位置５６１を通り終了する。

【００４４】上記のように、インテル互換性浮動小数点ユニットアーキテクチャは、スタックレジスタにアクセスするために１－オペランド命令或いは２－オペランド命令を用いる。これは、命令ソースの１つが命令ターゲットと同じであることを意味する。従ってソース値の１つが命令の実行により消失する。両方のソース値が命令の実行中に保存される必要がある場合には、その命令の実行により消失するようになる１つの値が一時的レジスタに複製されなければならない。他のコンピュータアーキテクチャは同様の特性を有し、同様の方法において取り扱われる。

【００４５】図１０は概ね参照番号６２０により示され、ＩＲ命令を変更し、レジスタスタックから無効疑似レジスタを除去するために用いられるスタック安定化プロセスを示す。ＩＲはなおも疑似レジスタを参照するが、ＩＲ命令は、命令実行後に疑似レジスタが無効である場合には、命令のスタックポップ別形態を用いるために変更されることができる。詳細には疑似レジスタの値は最後の使用後に無効になる。

【００４６】プロセス６２０は「開始」位置６２１で開始し、「無効値」決定手順６２３に続く。「無効値」決定手順６２３は、命令により参照される疑似レジスタが命令後に無効であるか否かを判定する。そうである場合には、プロセス６２０は、その命令が無効値をポップする別形態式或いはシーケンスを有するか否かを判定する「命令がポップ別形態所有」決定手順６２６に続く。命令により参照される疑似レジスタが命令後に無効でない場合には、プロセス６２０は「終了」位置６２７を通り終了する。

【００４７】「命令がポップ別形態所有」決定手順６２６が、命令がポップ別形態を有するものと判定する場合には、プロセス６２０は非ポップ命令別形態をポップ命令別形態に置き換える「命令別形態変更」手順６２９に続く。ポップ命令別形態が存在しない場合には、プロセス６２０はコードシーケンスを挿入し、無効疑似レジスタを明示的にポップする「ポップ命令シーケンス挿入」手順６２８に続く。プロセス６２０は「終了」位置６２７を通り終了する。

【００４８】ここでレジスタスタック状態の使用について議論することは有用である。レジスタスタック状態は、疑似レジスタとスタックレジスタとの間のマッピン

25

グを実現する。スタックレジスタがレジスタスタックとして構成されるので、他の値がレジスタスタックへプッシュ及びレジスタスタックからポップされるのに応じて、特定の値を含むスタックレジスタへのスタックオフセットが変化する。レジスタスタック状態は疑似レジスタをスタックレジスタにマップする。本発明の一態様は、レジスタスタックが変化するのに応じて、レジスタ＊

26

＊スタック内の値の位置を追跡することによるレジスタスタック状態のメンテナンスである。基本ブロックの一部を構成するｘ＝（ａ＋ｂ）＊（ａ＋ｄ）を考慮する。レジスタ割当て部からのＩＲコードは、表１と同様の内容を探索するであろう。

【００４９】

【表１】

表１

fld	a->pr1
fsto	pr1->pr2
fadd	pr1, b->pr1
fadd	pr2, d->pr2
fmul	pr2, pr1->pr2
fsto	pr2->x

【００５０】ここで「ｐｒｎ」は疑似レジスタ「ｎ」を示す。

【００５１】表２は、疑似レジスタがレジスタスタック状態を用いてスタックレジスタにいかに割り当てられる※

※かを示す。

20

【００５２】

【表２】

表２

Inst	Operands	F0	F1	F2	F…
fld	a->F0	pr1	pr3		?
fsto	F0->F2	pr1	pr3	pr2	?
fadd	F0, b->F0	pr1	pr3	pr2	?
fxch	F2, F0	pr2	pr3	pr1	?
fadd	F0, d->F0	pr2	pr3	pr1	?
fmul	F2, F0->F0	pr2	pr3	pr1	?
fstop	F0->X	pr3	pr1	?	?
…	…	…	…	…	…

【００５３】ここで「Ｆｎ」はスタックレジスタ「ｎ」を示し、「ｐｒｎ」は再び疑似レジスタ「ｎ」を示す。表２の右４段は、いずれの疑似レジスタが各スタックレジスタに割り当てられたかを示す。「ｐｒ３」は後続の動作に用いられる現存の疑似レジスタである。「ｐｒ２」は、「Ｘ」への格納後に無効であると見なされ、そのためレジスタスタックからポップされるということに注目されたい。レジスタスタック状態は、レジスタスタックの先頭からスタックレジスタへのオフセットを有する各疑似レジスタ間の関係を保持する。レジスタスタック状態は全ての命令に対して保持される。レジスタスタックの先頭からのオフセットを用いるために命令を変更することにより、疑似レジスタを用いる各命令のオペランドは、疑似レジスタにマップされるスタックレジスタをアドレス指定するように変更される。レジスタスタック状態に格納された各オフセットは、レジスタスタックがその命令により入れ替えられるのに応じて変化する。

40

50

【００５４】図１１は概ね参照番号７００により示され、疑似レジスタをレジスタスタック内のスタックレジスタにマップするための疑似レジスタマッピングプロセスを示す。プロセス７００は図４の「スタックレジスタをマップ」手順５５５により呼び出される。プロセス７００は「開始」位置７０１で開始し、「初期基本ブロックマッピング状態退避」手順７０３に続く。「初期基本ブロックマッピング状態退避」手順７０３は、レジスタスタック状態を初期化する。レジスタスタック状態は、スタックオフセット値を疑似レジスタに関連付けることにより、疑似レジスタをレジスタスタック内のスタックレジスタにマップする。「初期基本ブロックマッピング状態退避」手順７０３は、基本ブロックへの入口におけるレジスタスタック状態の複製を退避する。初期基本ブロックレジスタスタック状態が退避された後、プロセス

７００は「各命令繰返し」手順７０５に続く。「各命令繰返し」手順７０５は基本ブロック内の各命令を繰り返す。基本ブロック内の全ての命令が繰り返された後、プロセス７００は、その基本ブロックの終了時に存在するレジスタスタック状態をｆｓｔａｔｅ（ｂｌｏｃｋ－ｉｄ，ｏｕｔｐｕｔ）として退避する。「最終基本ブロックマッピング状態退避」手順７０７に続く。その後プロセス７００は「終了」位置７０９を通り終了する。

【００５５】「初期基本ブロックマッピング状態退避」手順７０３及び「最終基本ブロックマッピング状態退避」手順７０７により退避されるレジスタスタック状態を用いて、図１２に関連して後に記載されるように、基本ブロック間のレジスタスタック使用を正規化する。

【００５６】「各命令繰返し」手順７０５により繰り返される各命令は検査され、「浮動小数点レジスタ命令」決定手順７１１においてその命令が疑似レジスタにアクセスするか否かを判定する。繰返し済み命令が疑似レジスタにアクセスしない場合には、その命令は関連せず、プロセス７００は「各命令繰返し」手順７０５に戻り、次の命令を繰り返す。

【００５７】しかしながら繰返し済み命令が疑似レジスタにアクセスする場合には、プロセス７００は、繰返し済み命令が、レジスタスタック（及び関連するレジスタスタック状態）に対する入替えを必要とするか否かを判定する「スタック状態変更」決定手順７１３に続く。１つのそのような命令の一例は、演算される値がレジスタスタックの先頭スタックレジスタ内に格納されることが必要となるインテルＦＳＱＲＴ命令である。従って対象の値が任意の他のスタックレジスタ内に格納される場合には、そのレジスタスタックは、その対象の値がレジスタスタックの先頭に移動するように入れ替えられなければならない。「スタック状態変更」決定手順に７１３が、繰返し済み命令がスタック入替えを必要とすると判定する場合には、プロセス７００は「レジスタスタック入替え命令挿入」手順７１５に続く。「レジスタスタック入替え命令挿入」手順７１５は、レジスタスタックを繰返し済み命令に適した状態にするために命令を挿入する。インテル互換性ＦＰＵアーキテクチャ内部では、「レジスタスタック入替え命令挿入」手順７１５は、レジスタスタックを入れ替えるためにＦＸＣＨ命令を挿入する。次にレジスタスタック状態は「マッピング状態更新」手順７１７により更新される。こうしてレジスタスタック状態は、レジスタスタックのスタックレジスタの位置の変化に応答する。従ってコンパイラは、レジスタスタックにアクセスする各命令の動作に応じてレジスタスタックの状態を保持する。

【００５８】「スタック状態変更」決定手順７１３が、繰返し済み命令がスタック入替えを必要としないと判定する場合には、プロセス７００は「マッピング状態更新」手順７１７に続く。「マッピング状態更新」手順７１７は挿入された命令を検査し、その命令がレジスタスタックを変更する場合には、その手順がレジスタスタック状態への対応する変更を生成する。こうしてコンパイラは、レジスタスタックにアクセスする各命令の動作に応じてレジスタスタックの状態を保持する。

【００５９】一度レジスタスタック状態が更新されれば、プロセス７００は、レジスタスタック状態の情報を用いて、疑似レジスタからの繰返し済み命令のオペランドを、レジスタスタックへのスタックオフセットに変換する「疑似レジスタのレジスタスタックへのマップ」手順７１９に続く。こうして変換された命令は、疑似レジスタの代わりにレジスタスタックのスタックレジスタを参照する。一度繰返し済み命令が変換されれば、プロセス７００は、基本ブロック内の全ての命令が繰り返されるまで「各命令繰返し」手順７０５において命令を繰り返す。

【００６０】プロセス７００は、ＣＦＧの先頭から底への通過中に各命令ブロックに対して呼び出される。ＣＦＧの先頭から底への通過の終了時に、各命令ブロックに対する入力レジスタスタック状態及び出力レジスタスタック状態が退避されている。さらに疑似レジスタを用いる基本ブロック内の各命令は、ここでレジスタスタックへのオフセットを用いてレジスタスタックにアクセスする。

【００６１】残りのステップは、親基本ブロックからの出口におけるレジスタスタックの状態を、現在の基本ブロックの開始時点におけるレジスタスタックの状態に突き合わせることである。これは、「ＣＦＧの底から先頭への通過」手順５５７により実行されるＣＦＧの底から先頭への通過中にブロックを正規化することにより達成される。

【００６２】図１２は、概ね参照番号８００により示され、親基本ブロック（ｂｌｏｃｋ－ｊ）からの最後のレジスタスタック使用を現在の基本ブロック（ｂｌｏｃｋ－ｋ）の入力レジスタスタック使用に正規化する正規化命令を挿入する基本ブロックレジスタスタック正規化プロセスを示す。正規化は、いずれのレジスタが現存する入力ｂｌｏｃｋ－ｋであるが、ｂｌｏｃｋ－ｊからの出口には現存しないかを判定し、かついずれのレジスタがｂｌｏｃｋ－ｊからの出口に現存するが、ｂｌｏｃｋ－ｋには用いられないかを判定するプロセスである。プロセス８００は、「ＣＦＧの底から先頭への通過」手順５５７によりＣＦＧが底から先頭まで通過されるのに応じて、「基本ブロック正規化」手順５５９により呼び出される。プロセス８００は「開始」位置８０１で開始し、「新規の現存レジスタ判定」手順８０３に続く。「新規の現存レジスタ判定」手順８０３は、親ブロックからの出口レジスタスタック状態を、現在ブロックからの入口レジスタスタック状態と比較する。こうしてＣＦＧのｂｌｏｃｋ－ｊとｂｌｏｃｋ－ｋとの間の一致を探索する

場合に、新規の現存レジスタのセットは以下のように定義される。

【００６３】

【数１】ｐｕｓｈ（ｂｌｏｃｋ－ｊ，ｂｌｏｃｋ－ｋ）＝ｉｎｐｕｔ（ｂｌｏｃｋ－ｋ）－ｆｓｔａｔｅ（ｂｌｏｃｋ－ｊ，ｏｕｔｐｕｔ）同様にして、「親ブロックの無効レジスタ判定」手順８０５は、以下に定義されるような同じエッジにおける新規の無効レジスタのセットを判定する。

【００６４】

【数２】ｐｏｐ（ｂｌｏｃｋ－ｊ，ｂｌｏｃｋ－ｋ）＝ｆｓｔａｔｅ（ｂｌｏｃｋ－ｊ，ｏｕｔｐｕｔ）－ｉｎｐｕｔ（ｂｌｏｃｋ－ｋ）

「正規化要求」決定手順８０７は、ｂｌｏｃｋ－ｊ出口におけるレジスタスタック状態をｂｌｏｃｋ－ｋ入口におけるレジスタスタック状態に突き合わせるために必要とされるスタック入替え並びにプッシュ及びポップ値を検査する。ｂｌｏｃｋ－ｊとｂｌｏｃｋ－ｋとの間で正規化が必要とされる場合には、プロセス８００は「基本ブロック正規化」手順８０９に続く。「基本ブロック正規化」手順８０９は、レジスタスタックを操作し、ｂｌｏｃｋ－ｊ出口レジスタスタック状態が、図１３に関連して後に記載されるようにｂｌｏｃｋ－ｋ入力レジスタスタック状態と突き合わせられるようにする命令を挿入する。その後プロセス８００は「終了」位置８１１を通り終了する。

【００６５】しかしながらｂｌｏｃｋ－ｊとｂｌｏｃｋ－ｋとの間に正規化が必要とされない場合には、プロセス８００は「終了」位置８１１を通り終了する。

【００６６】図１３は、概ね参照番号８５０により示され、ｂｌｏｃｋ－ｊとｂｌｏｃｋ－ｋとの間のレジスタスタックを正規化する基本ブロックを挿入する正規化命令挿入プロセスを示す。プロセス８５０は、図１２の「基本ブロック正規化」手順８０９により呼び出される。プロセス８５０は「開始」位置８５１で開始し、新規の基本ブロックがＣＦＧに挿入されなければならないか否かを、或いは現存する基本ブロックにおいてレジスタスタック正規化が発生ことができるか否かを判定する「新規の基本ブロック要求」決定手順８５３に続く。新規の基本ブロックは、ｂｌｏｃｋ－ｋがｂｌｏｃｋ－ｊに対する唯一の後続ブロックである場合には必要とされない。この状況では、レジスタスタック正規化命令はｂｌｏｃｋ－ｊに直接挿入されることができる。またｂｌｏｃｋ－ｊがｂｌｏｃｋ－ｋに対する唯一の親である場合には、正規化命令は、ｂｌｏｃｋ－ｋの始めに直接挿入されることができる。この状況が当てはまらない場合には、正規化基本ブロックは、「正規化基本ブロック挿入」手順８５５においてＣＦＧに挿入される。正規化基本ブロックは、親基本ブロックから出るレジスタスタックを正規化し（入れ替え）、子基本ブロックへの入口において予想されるレジスタスタックと一致させる命令を含む。これらの状態が図１４及び図１５により示される。

【００６７】「レジスタスタックプッシュ命令挿入」手順８５７は、レジスタスタック上に新規の現存レジスタ（上記プッシュ定義により定義された）をプッシュする命令を挿入する。次に「レジスタスタックポップ命令挿入」手順８５９は、レジスタスタックをリオーダする命令及びレジスタスタックから無効レジスタをポップする命令を挿入する。その後「レジスタスタック入替え命令挿入」手順８６１は、ｂｌｏｃｋ－ｋに対する入力レジスタスタック状態に応じてレジスタスタックを配置する命令を挿入する。次にプロセス８５０は「終了」位置８６３を通り終了する。

【００６８】図１２及び図１３のプロセスはエッジにより接続される基本ブロックのレジスタスタック状態を用いて、基本ブロック間のレジスタスタックを正規化することは当業者には理解されよう。この正規化はレジスタスタックを入れ替え、レジスタスタックの現存のレジスタが、基本ブロックにおいて使用するためのレジスタスタックの適当なオフセット位置に存在できるようにする過程を含む。

【００６９】図１４は、概ね参照番号９００により示され、本発明により動作することができる制御フローグラフを示す。制御フローグラフ９００は、エッジ－ｊ／ｋ９０５によりｂｌｏｃｋ－ｋ９０３に接続されるｂｌｏｃｋ－ｊ９０１を含む。ｂｌｏｃｋ－ｋ９０３はエッジ－ｋ／ｍ９１０を用いてｂｌｏｃｋ－ｍ９０９に接続する。さらにｂｌｏｃｋ－ｌ９０７はエッジ－ｌ／ｍ９１１によりｂｌｏｃｋ－ｍ９０９に接続する。またｂｌｏｃｋ－ｋ９０３は、エッジ－ｋ／ｎ９１２を介してｂｌｏｃｋ－ｎ９１３に接続する。またｂｌｏｃｋ－ｌ９０７はエッジ－ｌ／ｎ９１５を用いてｂｌｏｃｋ－ｎ９１３に接続する。ｂｌｏｃｋ－ｍ９０９はエッジ－ｍ／ｏ９１８を用いてｂｌｏｃｋ－ｏ９１７に接続する。ｂｌｏｃｋ－ｏ９１７はエッジ－ｏ／ｐ９２０を用いてｂｌｏｃｋ－ｐ９１９に接続する。またｂｌｏｃｋ－ｎ９１３はエッジ－ｎ／ｏ９２１を用いてｂｌｏｃｋ－ｏ９１７に接続する。またｂｌｏｃｋ－ｏ９１７はエッジ－ｏ／ｑ９２５を用いてｂｌｏｃｋ－ｑ９２３に接続する。またｂｌｏｃｋ－ｑ９２３はエッジ－ｑ／ｒ９２８を用いてｂｌｏｃｋ－ｒ９２７に接続する。またｂｌｏｃｋ－ｒ９２７はエッジ－ｒ／ｒ９２９によりそれ自体に接続する。

【００７０】図１４を検証する場合に以下の点に注目されたい。ｂｌｏｃｋ－ｋ９０３はｂｌｏｃｋ－ｊ９０１に隣接する。ｂｌｏｃｋ－ｏ９１７は２つの親（ｂｌｏｃｋ－ｍ９０９及びｂｌｏｃｋ－ｎ９１３）及び２つの子（ｂｌｏｃｋ－ｐ９１９及びｂｌｏｃｋ－ｑ９２３）を有する。ｂｌｏｃｋ－ｍ９０９、ｂｌｏｃｋ－ｎ９１

３及びｂｌｏｃｋ－ｒ９２７はそれぞれ、ブロックの上部に入る２つのエッジを有する。

【００７１】図１５は、概ね参照番号９５０により示され、図１４の制御フローグラフ９００上で動作する図１２及び図１３により示されるプロセスの動作から生じる正規化された制御フローグラフを示す。制御フローグラフ９００は、親基本ブロックを出るスタックレジスタを、子基本ブロックに入るスタックレジスタに入れ替えるための正規化命令を含む基本ブロックを挿入するように変更される。図１２のプロセス８００を制御フローグラフ９００に適用することにより、以下のことがわかる。

【００７２】１）ｂｌｏｃｋ－ｋ９０３はｂｌｏｃｋ－ｊ９０１に隣接する。従って図１３のプロセス８５０により挿入されるレジスタスタック正規化命令はｂｌｏｃｋ－ｊ９０１（ｂｌｏｃｋ－ｋ９０３に対する親ブロック）内に挿入されることができる。以下の基本ブロックの組み合わせ、ｂｌｏｃｋ－ｍ９０９及びｂｌｏｃｋ－ｏ９１７、ｂｌｏｃｋ－ｎ９１３及びｂｌｏｃｋ－ｏ９１７並びにｂｌｏｃｋ－ｑ９２３及びｂｌｏｃｋ－ｒ９２７の場合に同様の状況が生じる。

【００７３】２）ｂｌｏｃｋ－ｏ９１７はｂｌｏｃｋ－ｐ９１９及びｂｌｏｃｋ－ｑ９２３の両方に対する親ブロックである。従ってレジスタスタック正規化命令（各子ブロックに適している）は、図１３のプロセス８５０により、ｂｌｏｃｋ－ｐ９１９及びｂｌｏｃｋ－ｑ９２３内に挿入される。

【００７４】３）一般的な場合に、ｂｌｏｃｋ－ｋ９０３及びｂｌｏｃｋ－ｌ９０７（並びにｂｌｏｃｋ－ｒ９２７のループエッジ）の子ブロックはそれぞれ、一般に正規化命令が任意の現存の基本ブロックに挿入されることができない（基本ブロックは実行フロー分岐を持たない）ため、親基本ブロックと子基本ブロックとの間のレジスタスタック状態を正規化するために、ＣＦＧに挿入される正規化基本ブロックを有する必要がある。従ってｂｌｏｃｋ－ｋ９０３とｂｌｏｃｋ－ｍ９０９との間のレジスタスタックを正規化するための命令を含むｂｌｏｃｋ－ｋ／ｍ９５１がエッジ－ｋ／ｍ９１０に挿入される。ｂｌｏｃｋ－ｋ／ｍ９５１は、ｂｌｏｃｋ－ｋ９０３を出るレジスタスタック状態により識別される構成からのレジスタスタックを変更し、ｂｌｏｃｋ－ｍ９０９に対する入力レジスタスタック状態を一致させる。ｂｌｏｃｋ－ｋ／ｎ９５３、ｂｌｏｃｋ－ｌ／ｎ９５５及びｂｌｏｃｋ－ｌ／ｍ９５７は、それぞれのエッジに対して相応する正規化を実現する。最終的にｂｌｏｃｋ－ｒ／ｒ９５９がエッジ－ｒ／ｒ９２９に挿入され、ｂｌｏｃｋ－ｒ９２７の出口レジスタスタック状態を、ｂｌｏｃｋ－ｒ９２７の入口レジスタスタック状態に正規化する。親出口レジスタスタック状態が、子ブロックの入口レジスタスタックに完全に一致する場合には、正規化ブロックはＣＦＧに挿入される必要はないということは当業者には理解されよう。

【００７５】本発明は、基本ブロック間のスタック正規化命令数を最小限にし、コード拡張を低減するということは当業者には理解されよう。

【００７６】本発明により、現存するレジスタ割当て及び最適化技術が、レジスタスタック内に構成されるスタックレジスタと共に用いられるようになるということは当業者には理解されよう。

【００７７】上記内容から、本発明は（限定するわけではないが）以下の利点を有することが明らかであろう。

【００７８】１）本発明はレジスタスタックにおけるレジスタスワッピングを最小限にする。

【００７９】２）本発明は基本ブロック間でレジスタスタック使用を正規化することから生じる、可能なコード拡張を最小限にする。

【００８０】本発明は好適な実施例に関連して記載されてきたが、本発明の範囲から逸脱することなく種々の変更例及び代替例が実現されることは当業者には理解されよう。従って本発明の範囲はここで議論された特定の本発明の実施例に制限されるわけではなく、添付の請求項及びその等価内容によってのみ画定されるべきである。

【００８１】

【発明の効果】上記のように、本発明により、従来のレジスタ割当て技術及び最適化技術を、レジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現することができる。

【図面の簡単な説明】

【図１】好適な実施例に従って本発明を用いることができるコンピュータシステムを示す。

【図２】好適な実施例によるコンパイラを示す。

【図３】好適な実施例によるスタックレジスタ置換えプロセスの概要を示す。

【図４】好適な実施例によるＩＲ３－オペランド命令変換プロセスの概要を示す。

【図５】好適な実施例による詳細なＩＲ３－オペランド命令変換プロセスを示す。

【図６】好適な実施例により、疑似レジスタからの命令アクセスをスタックレジスタに変換するためのプロセスを示す。

【図７】好適な実施例により、ＣＦＧにより表される現存疑似レジスタを最適化し、基本ブロックへの入口及び基本ブロックからの出口において現存疑似レジスタを確定するプロセスを示す。

【図８】好適な実施例により無効スタックレジスタをレジスタスタックから除去するためのプロセスの概要を示す。

【図９】好適な実施例により疑似レジスタをスタックレジスタにマップするためのプロセスの概要を示す。

【図１０】好適な実施例により無効疑似レジスタを基本

33

ブロックから除去するために用いられるスタック安定化プロセスを示す。

【図１１】好適な実施例により疑似レジスタをレジスタスタックのレジスタにマップするために用いられる詳細なプロセスを示す。

【図１２】好適な実施例による基本ブロックレジスタスタック正規化プロセスの概要を示す。

【図１３】好適な実施例による詳細な正規化命令挿入プロセスを示す。

【図１４】好適な実施例により動作する非正規化ＣＦＧの一部を示す。

【図１５】図１４のＣＦＧに好適な実施例を適用した結果生じる正規化制御フローグラフを示す。

【符号の説明】

１００　コンピュータ
１０１　プロセッサ
１０３　中央処理装置（ＣＰＵ）
１０５　メモリ
１０７　入力／出力（Ｉ／Ｏ）
１０９　キーボード
１１１　ディスプレイユニット
１１３　ディスク記憶ユニット
１１５　ＣＤ－ＲＯＭドライブユニット
１１７　ＣＤ－ＲＯＭ媒体
１１９　プログラム及びデータ
１２１　実行プログラム
１２３　レジスタスタック
２００　コンパイラアーキテクチャ
２０１　ソースプログラム
２０３　フロントエンド
２０５　中間コード表現
２０７　中間表現（ＩＲ）コード最適化部
２０９　最適化済み中間表現
２１１　コード生成部
２１３　アセンブラソース／オブジェクトコードモジュール
３００　スタックレジスタ置換えプロセス
３０１　開始位置
３０３　「入力データ生成」手順
３０５　「３－オペランド命令変換」手順
３０７　「絶対レジスタ参照をスタック相対参照に変換」手順
３０９　終了位置
４００　ＩＲ命令変換プロセス
４０１　開始位置
４０３　「各基本ブロック巡回」手順
４０５　「３－オペランドＩＲ命令変換」手順
４０７　終了位置
４２０　ＩＲ命令変換プロセス
４２１　開始位置

34

４２３　「ＩＲ命令繰返し」手順
４２５　終了位置
４２７　「命令指数部確定」手順
４２８　「関連命令」決定手順
４２９　「指数部選択」手順
４３１　「プログラム順（in-order）選択」手順
４３２　「基本変換」手順
４３３　「実行順序変換（out-of-order）、交換可能」選択手順
４３５　「オペランド順序交換」手順
４３７　「実行順序変換、交換不可、両方向ｏｐ」選択手順
４３９　「オペコードを逆向きオペコードに置換え」手順
４４１　「デフォルト」手順
４４３　「命令拡張」手順
５００　「絶対レジスタ参照をスタック相対参照に変換」プロセス
５０１　開始位置
５０３　「擬似レジスタについての情報収集」手順
５０５　「スタックから無効レジスタを除去」手順
５０７　「擬似レジスタをスタックレジスタ上にマップ」手順
５０９　終了位置
５１０　擬似レジスタ存続期間伝搬プロセス
５１１　開始位置
５１３　「ＣＦＧの擬似レジスタ最適化」手順
５２１　終了位置
５３０　「スタックから無効レジスタ除去」プロセス
５３１　開始位置
５３３　「各基本ブロック巡回」手順
５３５　「基本ブロックの各関連命令繰返し」手順
５３６　「スタック安定化」手順
５３７　終了位置
５５０　「疑似レジスタアドレス指定をレジスタスタックアドレス指定に変換」プロセス
５５１　開始位置
５５３　「先頭から底までＣＦＧを通過」手順
５５５　「スタックレジスタをマップ」手順
５５７　「底から先頭までＣＦＧを通過」手順
５５９　「基本ブロック正規化」手順
５６１　終了位置
６２０　スタック安定化プロセス
６２１　開始位置
６２３　「無効値」決定手順
６２６　「命令がポップ別形態所有」決定手順
６２７　終了位置
６２８　「ポップ命令シーケンス挿入」手順
６２９　「命令別形態変更」手順
７００　擬似レジスタマッピングプロセス

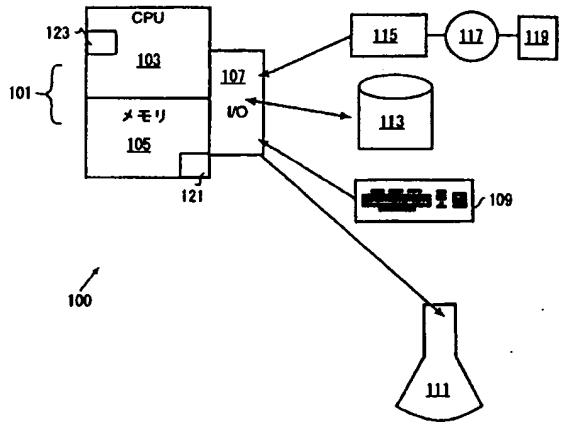
35

７０１　開始位置
７０３　「初期基本ブロックマッピング状態退避」手順
７０５　「各命令繰返し」手順
７０７　「最終基本ブロックマッピング状態退避」手順
７０９　終了位置
７１１　「浮動小数点レジスタ命令」決定手順
７１３　「スタック状態変更」決定手順
７１５　「レジスタスタック入替え命令挿入」手順
７１７　「マッピング状態更新」手順
７１９　「擬似レジスタのレジスタスタックへのマップ」手順
８００　基本ブロックレジスタスタック正規化プロセス
８０１　開始位置
８０３　「新規の現存レジスタ判定」手順
８０５　「親ブロックの無効レジスタ判定」手順
８０７　「正規化要求」決定手順
８０９　「基本ブロック正規化」手順
８１１　終了位置
８５０　正規化命令挿入プロセス
８５１　開始位置
８５３　「新規の基本ブロック要求」決定手順
８５５　「正規化ブロック挿入」手順
８５７　「レジスタスタックプッシュ命令挿入」手順
８５９　「レジスタスタックポップ命令挿入」手順
８６１　「レジスタスタック入替え命令挿入」手順
８６３　終了位置

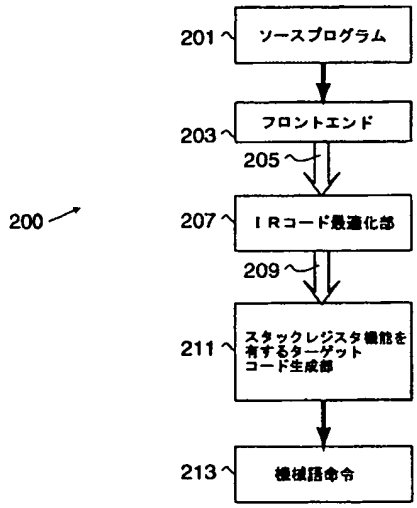
36

９００　制御フローグラフ
９０１　ｂｌｏｃｋ－ｊ
９０３　ｂｌｏｃｋ－ｋ
９０５　エッジ－ｊ／ｋ
９０７　ｂｌｏｃｋ－ｌ
９０９　ｂｌｏｃｋ－ｍ
９１０　エッジ－ｋ／ｍ
９１１　エッジ－ｌ／ｍ
９１２　エッジ－ｋ／ｎ
９１３　ｂｌｏｃｋ－ｎ
９１５　エッジ－ｌ／ｎ
９１７　ｂｌｏｃｋ－ｏ
９１８　エッジ－ｍ／ｏ
９１９　ｂｌｏｃｋ－ｐ
９２０　エッジ－ｏ／ｐ
９２１　エッジ－ｎ／ｏ
９２３　ｂｌｏｃｋ－ｑ
９２５　エッジ－ｏ／ｑ
９２７　ｂｌｏｃｋ－ｒ
９２８　エッジ－ｑ／ｒ
９２９　エッジ－ｒ／ｒ
９５１　ｂｌｏｃｋ－ｋ／ｍ
９５３　ｂｌｏｃｋ－ｋ／ｎ
９５５　ｂｌｏｃｋ－ｌ／ｎ
９５７　ｂｌｏｃｋ－ｌ／ｍ
９５９　ｂｌｏｃｋ－ｒ／ｒ

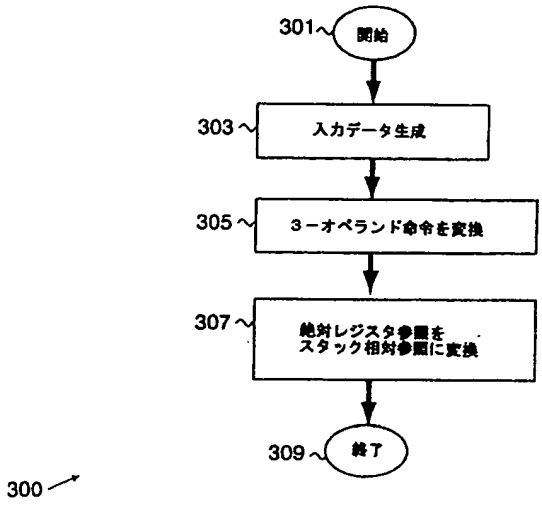
【図１】



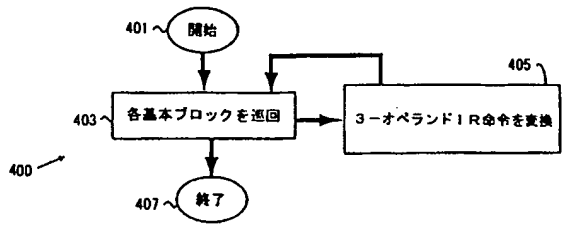
【図２】



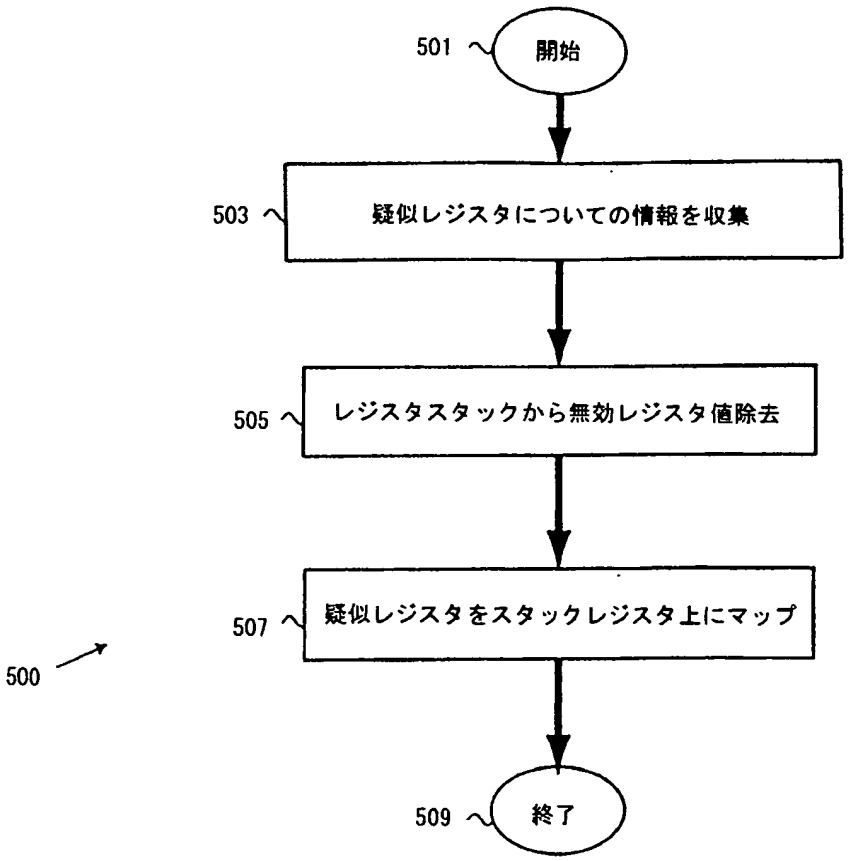
【図３】


301
開始
303
入力データ生成
305
３－オペランド命令を変換
307
絶対レジスタ参照を
スタック相対参照に変換
309
終了
300

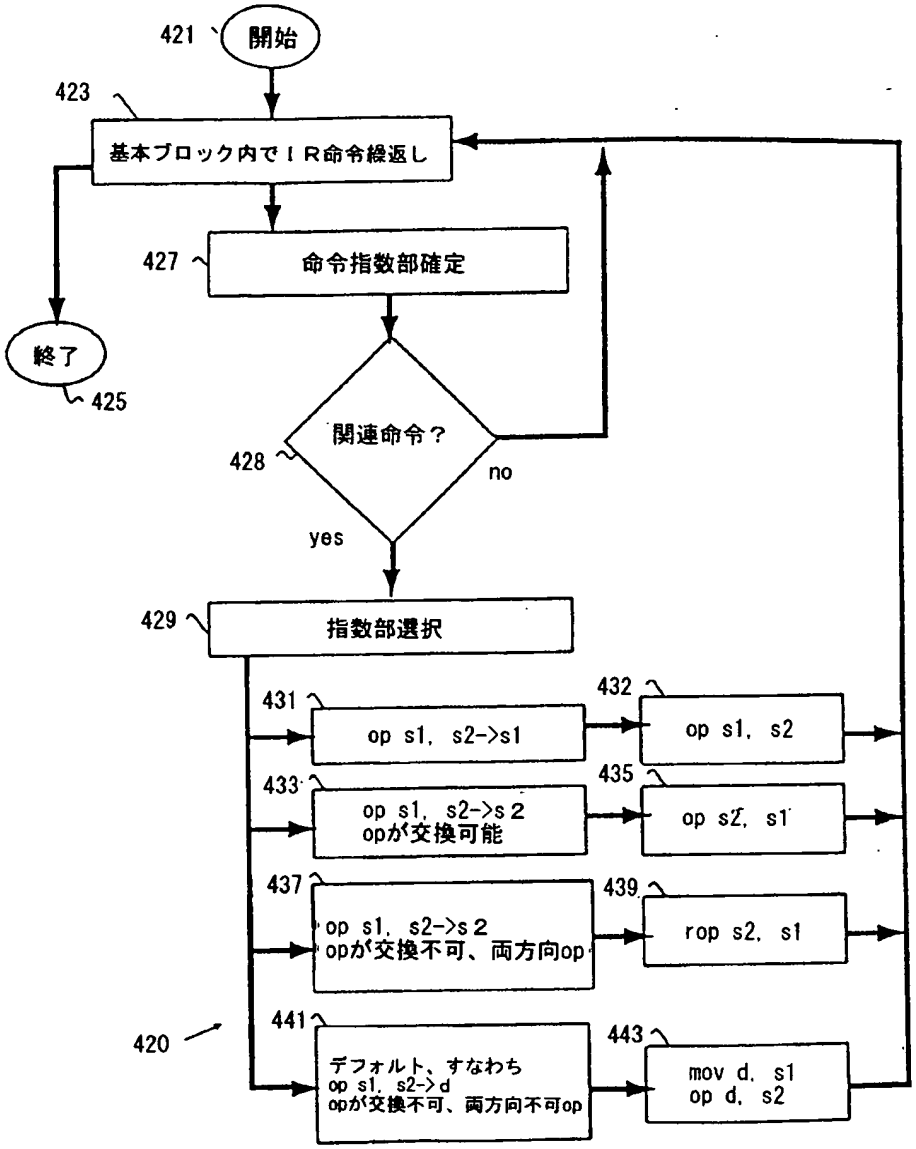
【図４】


401
開始
403
各基本ブロックを巡回
405
３－オペランドＩＲ命令を変換
407
終了
400

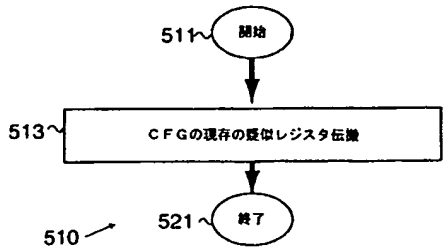
【図６】


501
開始
503
疑似レジスタについての情報を収集
505
レジスタスタックから無効レジスタ値除去
507
疑似レジスタをスタックレジスタ上にマップ
509
終了
500

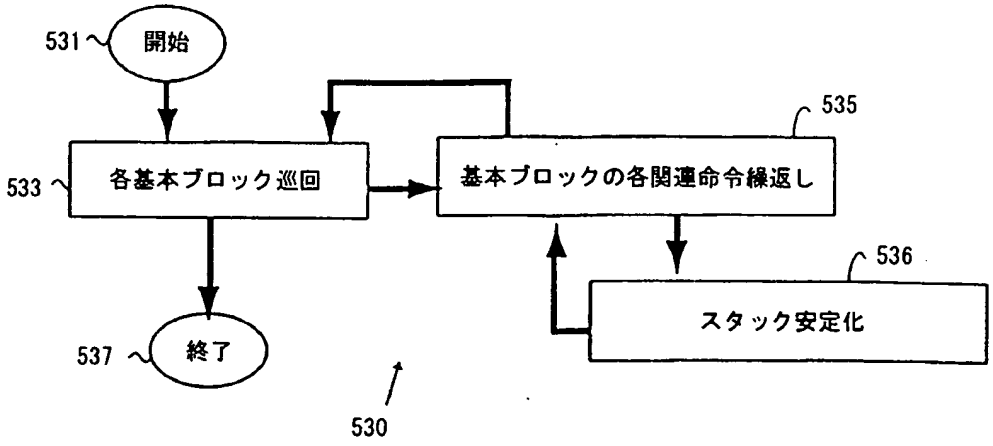
【図５】


421
開始
423
基本ブロック内でＩＲ命令繰返し
427
命令指数部確定
終了
425
関連命令？
428
no
yes
429
指数部選択
431
op s1, s2->s1
432
op s1, s2
433
op s1, s2->s２
opが交換可能
435
op s2, s1
437
op s1, s2->s２
opが交換不可、両方向op
439
rop s2, s1
441
デフォルト、すなわち
op s1, s2->d
opが交換不可、両方向不可op
443
mov d, s1
op d, s2
420

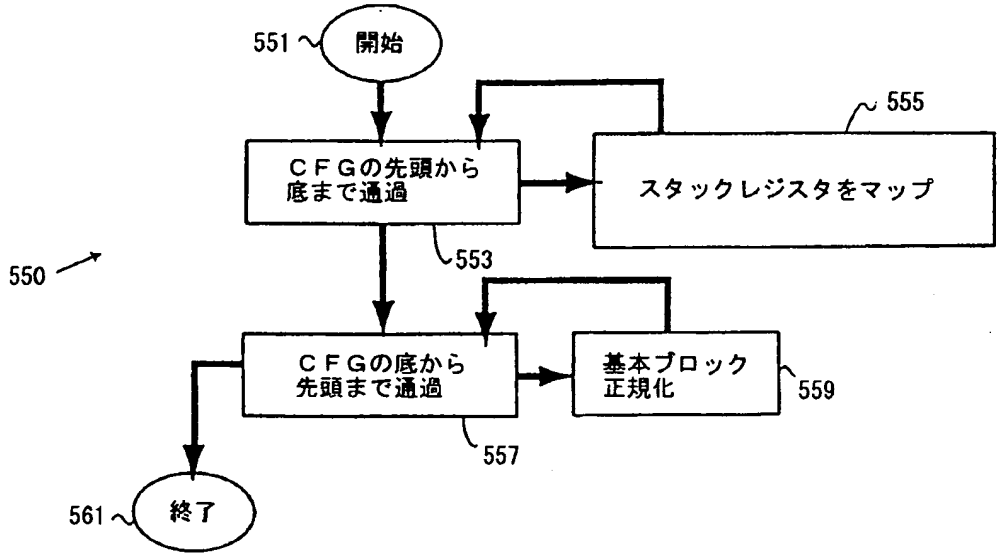
【図７】


511
開始
513
ＣＦＧの現存の擬似レジスタ伝搬
510
521
終了

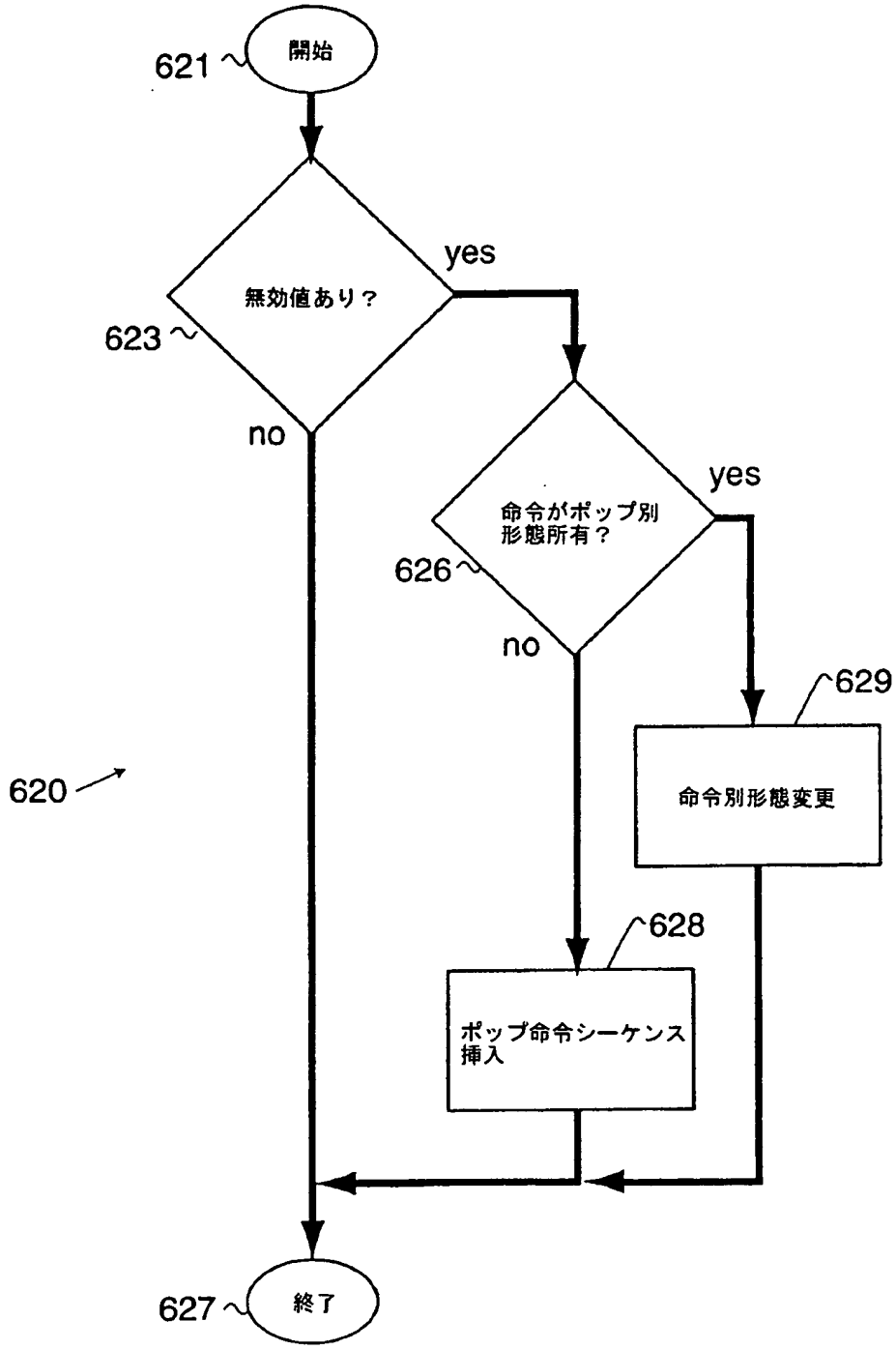
【図８】


531
開始
533
各基本ブロック巡回
535
基本ブロックの各関連命令繰返し
536
スタック安定化
537
終了
530

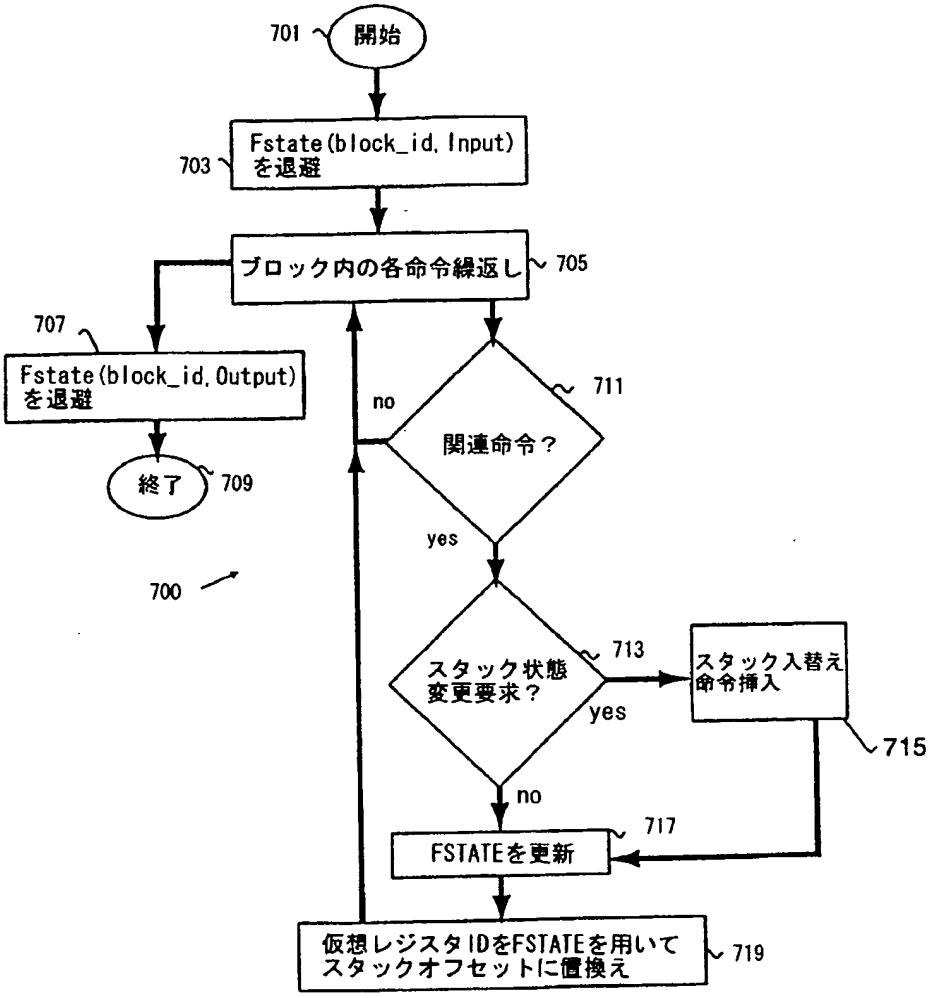
【図９】


551
開始
555
ＣＦＧの先頭から
底まで通過
スタックレジスタをマップ
550
553
ＣＦＧの底から
先頭まで通過
基本ブロック
正規化
559
557
561
終了

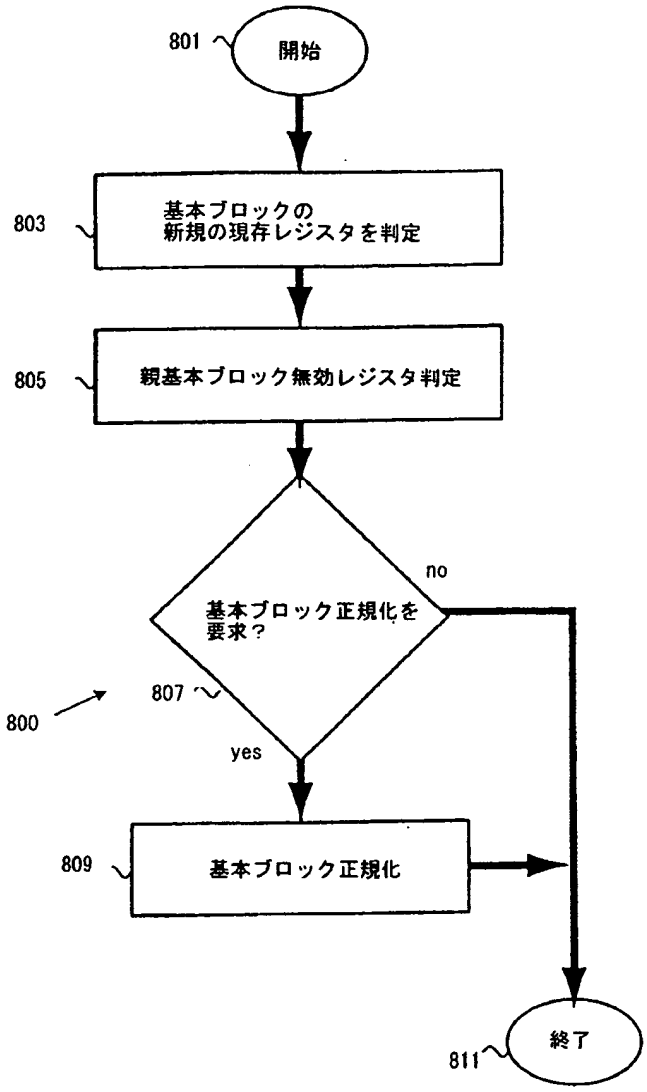
【図１０】


開始
621
無効値あり？
623
yes
no
命令がポップ別
形態所有？
626
yes
no
629
命令別形態変更
628
ポップ命令シーケンス
挿入
620
終了
627

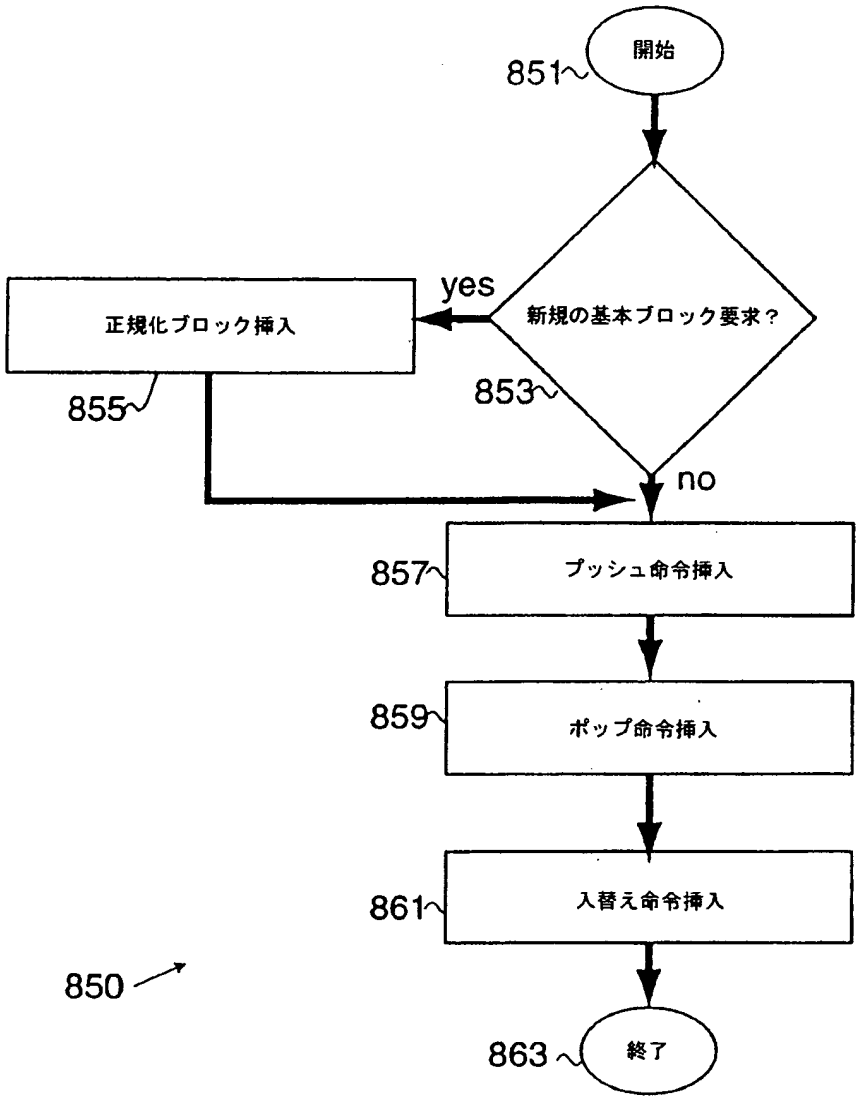
【図１１】


701
開始
703
Fstate(block_id, Input)
を退避
ブロック内の各命令繰返し
705
707
Fstate(block_id, Output)
を退避
終了
709
700
no
711
関連命令？
yes
713
スタック状態
変更要求？
yes
スタック入替え
命令挿入
715
no
717
FSTATEを更新
仮想レジスタIDをFSTATEを用いて
スタックオフセットに置換え
719

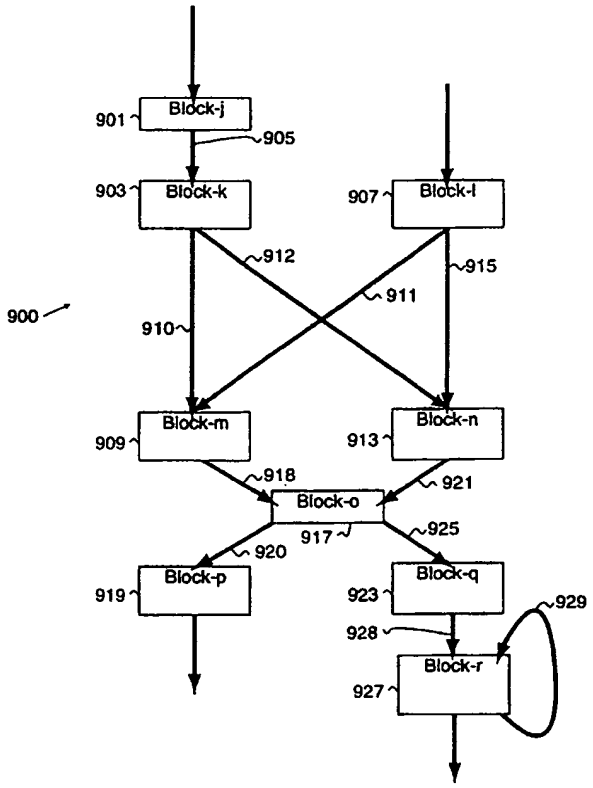
【図１２】


801
開始
803
基本ブロックの
新規の現存レジスタを判定
805
親基本ブロック無効レジスタ判定
基本ブロック正規化を
要求？
no
807
800
yes
809
基本ブロック正規化
811
終了

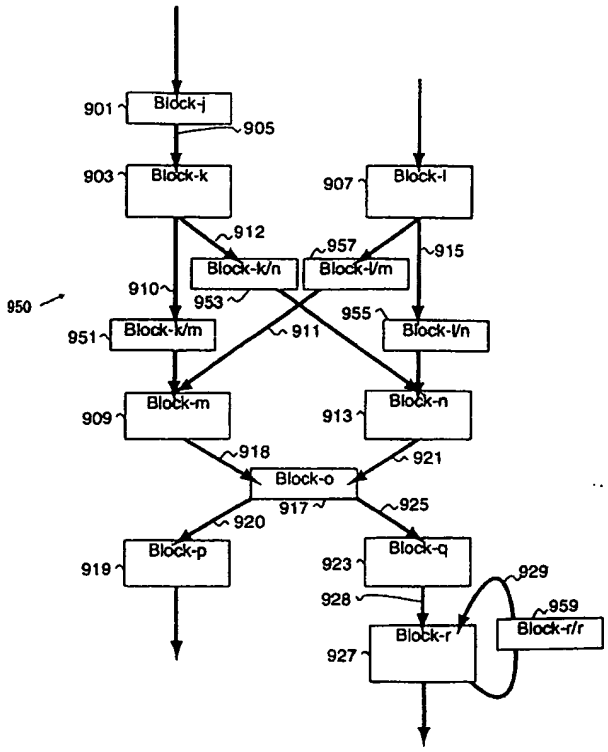
【図１３】


開始
851
新規の基本ブロック要求？
yes
no
853
正規化ブロック挿入
855
857
プッシュ命令挿入
859
ポップ命令挿入
861
入替え命令挿入
863
終了
850

【図１４】



【図１５】



フロントページの続き

(72)発明者　セルゲイ・ブイ・モロゾフ
ロシア630057ノボシビルスク・232・ペチャトニコフストリート　9
(72)発明者　ディビッド・エイ・セバーガー
アメリカ合衆国カリフォルニア州94550・リバーモア・シャルドネーウェイ　2271
(72)発明者　ディビッド・アール・ウォレス
アメリカ合衆国カリフォルニア州94133・サンフランシスコ・ジョーンズストリート　1960
(72)発明者　セルゲイ・エル・ウェニツキー
ロシア630057ノボシビルスク・57・ペチャトニコフストリート　9

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☑ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.